U.S.N.A. --- Trident Scholar project report; no. 342 (2006)

USING NON-ORTHOGONAL IRIS IMAGES FOR IRIS RECOGNITION

by

MIDN 1/C Ruth Mary Gaunt, Class of 2006
United States Naval Academy
Annapolis, MD

_____
(signature)

Certification of Adviser's Approval

Assistant Professor Robert W. Ives
Electrical Engineering Department

_____
(signature)

_____
(date)

Professor Delores M. Etter
Electrical Engineering Department

_____
(signature)

_____
(date)

Acceptance for the Trident Scholar Committee

Professor Joyce E. Shade
Deputy Director of Research & Scholarship

_____
(signature)

_____
(date)

USNA-1531-2

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATE COVERED
	5 May 2006	

4. TITLE AND SUBTITLE
Using non-orthogonal iris images for iris recognition

5. FUNDING NUMBERS

6. AUTHOR(S)
Gaunt, Ruth Mary, 1984-

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)		8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
US Naval Academy Annapolis, MD 21402		Trident Scholar project report no. 342 (2006)

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
This document has been approved for public release; its distribution is UNLIMITED.	

13. ABSTRACT The iris is the colored portion of the eye that surrounds the pupil and controls the amount of light that can enter the eye. The variations within the patterns of the iris are unique between eyes, which allows for accurate identification of an individual. Current commercial iris recognition algorithms require an orthogonal image of the eye (subject is looking directly into a camera) to find circular inner (pupillary) and outer (limbic) boundaries of the iris. If the subject is looking away from the camera (non-orthogonal), the pupillary and limbic boundaries appear elliptical, which a commercial system may be unable to process. This elliptical appearance also reduces the amount of information that is available in the image used for recognition. These are major challenges in non-orthogonal iris recognition. This research addressed these issues and provided a means to perform non-orthogonal iris recognition. All objectives set forth at the start of this project were accomplished. The first major objective of this project was to construct a database of non-orthogonal iris images for algorithm development and testing. A collection station was built that allows for the capture of iris images at 0° (orthogonal), 15°, 30°, and 45°. During a single collection on an individual, nine images were collected at each angle for each eye. Images of approximately 90 irises were taken, with 36 images collected per eye. Sixty irises were evaluated twice, resulting in a total of almost 7100 images in the database. The second major objective involved modifying the Naval Academy's one-dimensional iris recognition algorithm so it could process non-orthogonal iris images. An elliptical-to-circular (affine) transformation was applied to the non-orthogonal images to create circular boundaries. This permitted the algorithm to be run as designed, with this modified algorithm used in the recognition testing phase of the project. To evaluate the performance of the recognition algorithm and the feasibility of nonorthogonal recognition, rank-matching curves were generated. In addition, the accuracy of the database collection was evaluated by analyzing the iris boundary parameters of the nonorthogonal irises. MATLAB® software and the Naval Academy's biometric signal processing laboratory equipment were used to analyze the data and to implement this research, respectively.

14. SUBJECT TERMS	15. NUMBER OF PAGES
iris recognition ; iris images ; non-orthogonal recognition ; biometric signal processing	90
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

Abstract:

The iris is the colored portion of the eye that surrounds the pupil and controls the amount of light that can enter the eye. The variations within the patterns of the iris are unique between eyes, which allows for accurate identification of an individual. Current commercial iris recognition algorithms require an orthogonal image of the eye (subject is looking directly into a camera) to find circular inner (pupillary) and outer (limbic) boundaries of the iris. If the subject is looking away from the camera (non-orthogonal), the pupillary and limbic boundaries appear elliptical, which a commercial system may be unable to process. This elliptical appearance also reduces the amount of information that is available in the image used for recognition. These are major challenges in non-orthogonal iris recognition. This research addressed these issues and provided a means to perform non-orthogonal iris recognition. All objectives set forth at the start of this project were accomplished.

The first major objective of this project was to construct a database of non-orthogonal iris images for algorithm development and testing. A collection station was built that allows for the capture of iris images at 0˚ (orthogonal), 15˚, 30˚, and 45˚. During a single collection on an individual, nine images were collected at each angle for each eye. Images of approximately 90 irises were taken, with 36 images collected per eye. Sixty irises were evaluated twice, resulting in a total of almost 7100 images in the database.

The second major objective involved modifying the Naval Academy's one-dimensional iris recognition algorithm so it could process non-orthogonal iris images. An elliptical-to-circular (affine) transformation was applied to the non-orthogonal images to create circular boundaries.

This permitted the algorithm to be run as designed, with this modified algorithm used in the recognition testing phase of the project.

To evaluate the performance of the recognition algorithm and the feasibility of non-orthogonal recognition, rank-matching curves were generated. In addition, the accuracy of the database collection was evaluated by analyzing the iris boundary parameters of the non-orthogonal irises. MATLAB® software and the Naval Academy's biometric signal processing laboratory equipment were used to analyze the data and to implement this research, respectively.

Acknowledgments:

Thank you to the following individuals who have given so much of their time to help contribute to the success of this project:

Dr. Robert Ives, Electrical Engineering Department, USNA – Primary Project Adviser

Dr. Delores Etter, Electrical Engineering Department, USNA –Secondary Project Adviser

Dr. Lauren Kennell, Electrical Engineering Department, USNA-Research Asst. Professor

LT Robert Schultz, USN, Electrical Engineering Department, USNA

Mr. Jerry Ballman, Electrical Engineering Department, USNA- Laboratory Technician

Mr. Michael Wilson, Electrical Engineering Department, USNA – Laboratory Technician

Mr. Jeffery Dunn, National Security Agency – Chief, R3B

Dr. David Murley, National Security Agency – Lead Scientist, R3B

Mr. Robert Kirchner, National Security Agency – General Engineer, R3B

Mr. David Smith, Science Application International Corporation – Laboratory Technician

Ms. Janice Atwood, Booz Allen Hamilton– National Security Agency Liaison

Dr. James Matey, Sarnoff Corporation

Trident Scholar Committee Members

Table of Contents:

List of Figures:

List of Tables:

I. Introduction

Biometrics is the science that uses the distinct physical or behavioral traits of individuals to positively identify them. A wide variety of traits can be used, including the fingerprint, iris (see Fig. 1), face, hand geometry, voice, and even gait. Algorithms are developed to measure

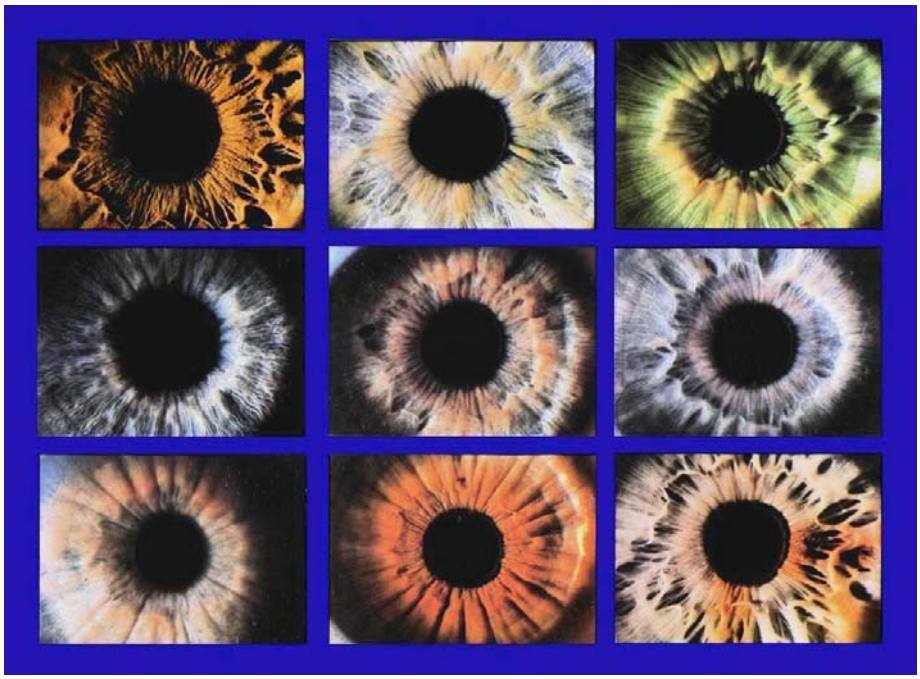

Figure 1. Collage of nine different iris images.
<http://www.cl.cam.ac.uk/users/jgd1000/iriscollage.jpg>

and quantize these various characteristics so they can be compared to a database of stored information in order for recognition to occur. This eliminates the need for passwords and personal identification numbers, which are easier to spoof than an individual's biometric information. Application of biometric technology increases confidence that only those people who are authorized to gain access to a particular resource or secure facility are able to do so.

Two of the major applications of biometrics are *verification* and *identification*. *Verification* is determining if individuals are who they say they are (a one-to-one comparison), and *identification* is determining if an individual is one of a number of known people in a

database (a one-to-many comparison), usually to allow access to a secure facility or network [2]. In addition to verification and identification, another important application is creating a "watchlist" or database of individuals of interest (e.g. known felons or terrorists) and scanning a high traffic area (such as national borders or airports) in the hopes of detecting one of the individuals on the list if they pass through (a many-to-many comparison). This is the most complex application of biometrics because it requires collecting biometric data on each person passing a "checkpoint" and comparing their features to all those on the watchlist: this can involve very large databases and many comparisons [2].

Another reason why the "watchlist" requires such a complex algorithm for identifying individuals is that the large-area scanning is done typically under covert conditions, where the subjects do not know that they are being observed [2]. At the present time, most biometric identification occurs when a subject knowingly approaches a biometric data collecting device, such as an iris or fingerprint scanner, and purposely presents the data necessary for identification, whether it be staring straight into a camera from a distance of only several inches or placing a finger on a fingerprint scanner. Currently, the data collection takes a noticeable amount of time as well, so the subject must also keep the observed biometric in proximity to the sensor until identification is completed. Once the biometric is collected, it also takes time for the collected data to be processed and compared to the database before a decision is made.

Today's biometrics applications require a cooperative subject and many controlled variables (such as proper illumination and distance to the sensor) for positive identification of an individual to occur. Decreasing the number of controlled variables requires significantly more complex algorithms. In the case of iris recognition, one of the variables that cannot be controlled

during covert observation is whether the collected image is orthogonal (eye looking directly into the camera) or non-orthogonal (off-axis) to the camera, as well as the orientation from an off-axis angle. Positive identification based on non-orthogonal iris images (see Fig. 2) is the problem that has been investigated as a part of this project. This includes development of a database of various off-axis iris images taken from different angles and development of an algorithm to match an off-axis iris to an iris in the database.

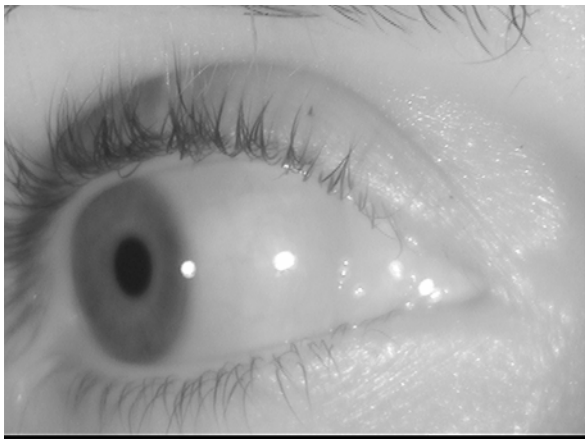

Figure 2. A Non-Orthogonal (Off-Axis) Iris Image.

II. Background

The iris is the only internal human organ that can be observed from the external environment, which is one reason why the iris is such a popular biometric. It is an area of tissue that lies behind the cornea and is responsible for controlling the amount of light that is able to enter the pupil as well as determining the eye color of an individual [3]. Iris pigmentation is caused by melanin, the same material that causes pigmentation in the skin [3]. Brown eyes are colored with eumelanin, and blue and green eyes are colored with pheomelanin [3].

Besides the functional capability of the iris, it has distinct physical features which are unique to each individual. In fact, a person's right and left irises do not share the exact same physical characteristics [3]. Iris patterns are determined by the four layers that make up the iris,

including the anterior border layer, the stroma, the dilator pupillae muscle, and the posterior pigment epithelium [3]. The combination of these four layers produces striations, freckles, pits, filaments, rings, and dark spots in addition to pigmentation [3]. An iris's patterns stabilize by the time a person is one year of age and remains constant throughout the person's lifetime unless damage to the eye occurs that would change the iris's unique patterns [3]. It is these patterns that are measured and quantified in an iris recognition system. These patterns tend to stand out more under near-infrared (NIR) illumination (approximately 790 nm wavelength), so most iris systems use an NIR camera.

The process of iris recognition can be broken down into five distinct steps that each requires special hardware or an algorithm to perform its function (see Fig. 3). The first step is to acquire an image of the iris. This is done with a NIR camera [4]. A frame grabber board can be used to capture a frame from the live video and bring it into a computer for further processing. The frame grabber serves as an interface between the analog video source and the PC being used during the collection process.

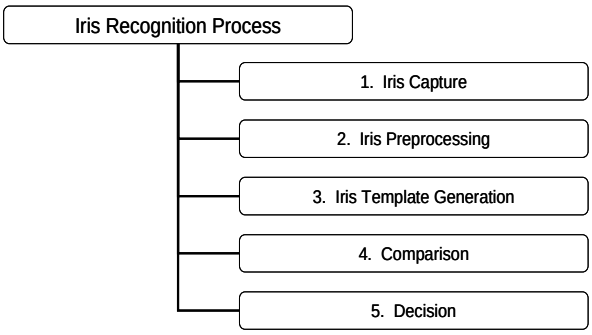

Figure 3. Iris Recognition Process.

The next step of iris recognition is the preprocessing of the iris [4]. Since the iris and pupil are approximately circular in shape (for an orthogonal image), this includes detecting the assumed circular pupil and converting the iris image from rectangular to polar coordinates (with the center of the pupil as the origin) so that the limbic (outer) boundary is virtually horizontal (see Fig. 4) [5]. Effects of

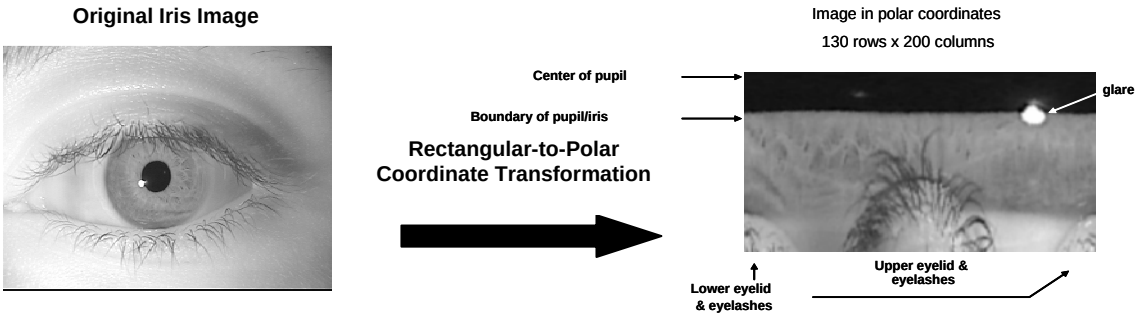

Figure 4. Rectangular-to-polar coordinate transformation.

glare and eyelashes are then accounted for by determining if any pixel values are outliers and removing them [5]. In addition, iris size can vary greatly due to the amount of light in the environment, which causes the pupil to constrict or dilate as the image is captured, and can also be affected by the distance from a person's face to the camera [5]. The preprocessing of the iris accounts for the iris size issues by normalizing the iris to a constant distance (number of pixels) between the pupillary and limbic boundaries, typically between 55 and 70 pixels [5]. Once the iris pixels are found, the rest of the image is discarded.

The third step in the iris recognition process is the generation of a method to store iris data in the database that offers an efficient and accurate way to identify individuals. This is usually called a "template" [4]. One method to do this was published by Du *et al* [5], in which local texture patterns (LTPs) are produced in order to eliminate any grayscale variation in the image due to different illumination conditions. Iris pixel values are replaced by LTP values to create an LTP image. Next, each row of pixels in the LTP image is then averaged by the iris template generation process in order to create a one-dimensional (1-D) template for a particular iris image. Since the top and bottom three rows (corresponding to the area of the iris closest to the pupil and farthest from the pupil, respectively) of the LTP generally are noisy, due to

inaccuracies in the actual detection of the boundaries, they are not considered when creating the iris template. In order for an iris to be identified, it must be enrolled in the database system, which usually requires multiple images of the same iris to be processed into an enrolled template and stored in the database.

Comparison of the iris templates in the database to the template produced by the presented iris is the next step in iris recognition [4]. In order to compare these templates in the 1-D algorithm, the Du measure is used [5]. This measurement computes the similarity of two 1-D templates (vectors), and takes into account the magnitude difference between the two templates and the angle between the two templates as though they were multi-dimensional vectors. The smaller the Du measure, the closer the two templates are to each other.

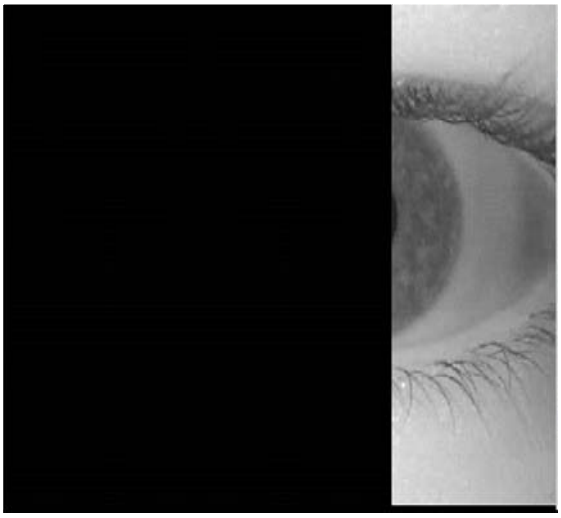

Figure 5. Partial Iris Recognition Testing.

Finally, after the presented iris template is compared to the iris templates in the database, a decision must be made from the results of the comparison [4]. This system outputs the closest *n* matches from the database as calculated by the smallest *n* Du measurements (*n* is chosen by the user) [5].

III. Previous Research of Partial Iris Recognition

While non-orthogonal iris recognition is not currently a viable means of recognition, Du *et al.* tested the 1-D recognition algorithm to see if recognition would work with just portions of an

orthogonal iris (see Fig. 5) [7]. It was found that partial iris recognition can work (with lower recognition performance), and the results show how likely a person is of being recognized when only a certain percentage of the iris is being used for recognition. The Institute of Automation, Chinese Academy of Sciences (CASIA) database (768 images of 108 irises) and the USNA orthogonal database (approximately 1500 images) were used to test the feasibility of partial iris recognition. The results show that when only 50% of the iris is being used for recognition, there is a 50% chance that the correct iris is ranked as the top choice for recognition and an 80% chance that it is ranked as one of the top five closest matches [7]. These results make it reasonable to assume that non-orthogonal iris recognition will achieve similar results because it has the same problem of not having the entire iris available for template generation and comparison.

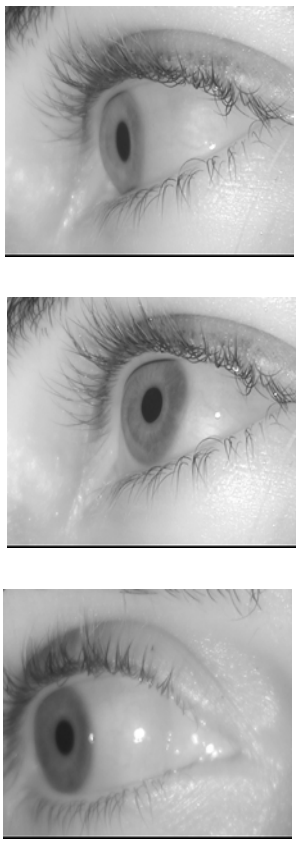

Figure 6. Examples of Partial Iris Images.

IV. Project Description

Despite its high recognition rate, one of iris recognition's major weaknesses is that it requires that the users be cooperative when making sure their eye is close enough to the camera and is still enough for a high quality iris image to be collected. In fact, current commercial systems require the iris to be orthogonal (or nearly so) to the camera, since their recognition algorithm must first detect the pupil, which is assumed to be a circle. This is only the case if the eye is looking directly at the camera lens. This makes it difficult or impossible for identification to

occur if the image is taken from an off-axis angle (see Fig. 6). The main purpose of this project has been to create a database of iris images taken from different off-axis angles (0°, 15°, 30°, 45°) using a near-infrared camera and to use these images to develop an algorithm to correctly identify an individual when presented with an off-axis image of the iris.

Figure 7. Non-Orthogonal Iris Image Collection Station.

One of the problems with non-orthogonal iris recognition is that when a person is not looking directly into the camera, the entire iris is not visible in the image because the iris is actually three-dimensional. Although the inner and outer boundaries of the iris can be located and the iris pixels can be extracted, information is missing in non-orthogonal iris images that is present in orthogonal images where all of the iris is visible. In order to complete the recognition algorithm, a procedure for the manipulation of the iris pixels in partial iris images is required in order achieve success in recognizing a individuals from their iris patterns when they are not staring directly into the camera.

V. Database Construction

In order to accurately collect non-orthogonal iris images at known orientation angles, a collection station has been built so the user's head remains stationary and the iris camera moves around the user's head (see Fig. 7).

The database of non-orthogonal iris images contains images taken at four known orientation angles: 0° (orthogonal), 15°, 30°, and 45°. First, the user places his or her chin in the chin rest so that the head remains stationary throughout the collection process. The chin rest can be raised and lowered so that no matter what the proportions of a person's face are, the eye can always be positioned to be in the center of the camera lens. Two thin metal rods are placed at the opposite end of the collection station for the user to focus on so that the only angle variation that occurs during the collection process is due to changes in camera position and not the shifting of the users' eyes. The camera is on a raised platform that moves on a track. It is held in place by a pin that fits into holes drilled at the desired collection angles for each eye. In addition, the collection station was constructed so that the distance from the camera to the eye is five inches, which is the desirable distance for achieving an optimal level of focus so that enough iris pattern information is available in each image. The high-quality, near-infrared camera used is from the LG IrisAccess® 3000 entry control system.

An existing iris collection graphical user interface [6] was altered so that information such as the angle at which the image is obtained is stored along with other information about the individual when the iris image is saved (see Fig. 8). This information includes user subject number, which eye is being collected (right or left), gender, iris color, iris age, and

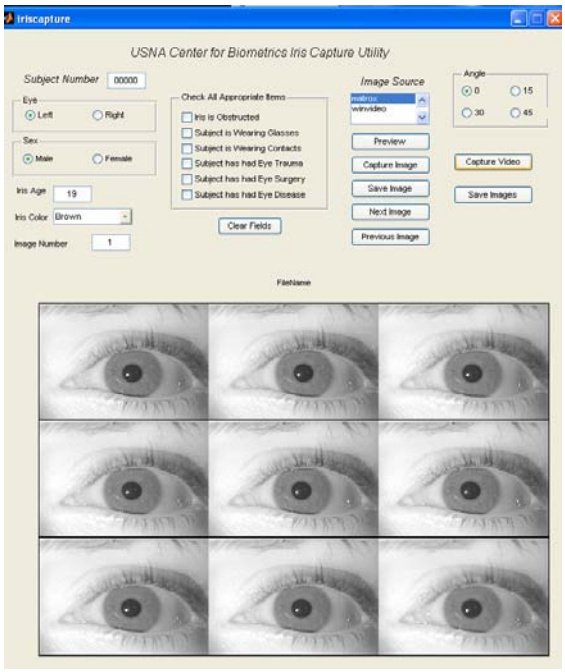

Figure 8. Graphical User Interface for Iris Image Collection.

whether the individual is wearing glasses, contacts, and if the user has a history of eye trauma or eye surgery [6]. For purposes of this research, users are instructed to remove their glasses so that changes in iris patterns due to optical distortion by glass lenses is not a variable.

The iris camera is used in conjunction with the MATLAB Image Acquisition and Image Processing Toolboxes and the Matrox Meteor II frame grabber to collect the data [6]. They are used to perform analog-to-digital conversion and to capture 9 images per second. These nine images are then saved on the computer. This means that for each eye, thirty-six images are obtained, since there are four different orientation angles and nine images are saved for each of these four angles. Figure 9 shows examples of images from each of the four orientation angles.

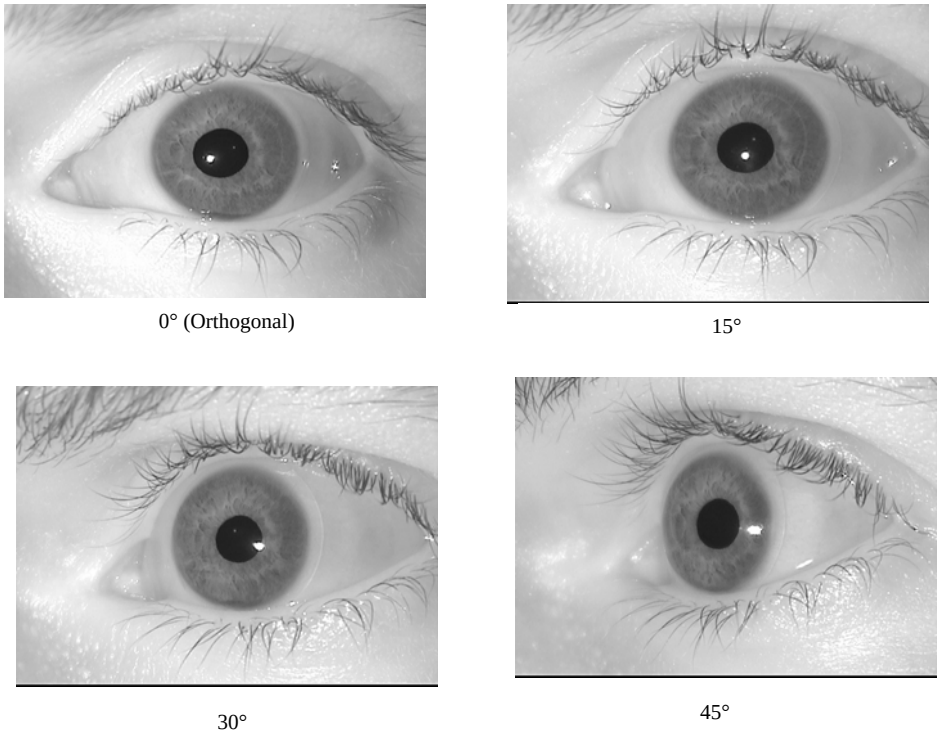

Figure 9. Samples of Database Images from Each Non-Orthogonal Angle.

Data for 94 irises is stored in the non-orthogonal database. These irises were collected at each non-orthogonal angle (as well as at 0°), and 58 went through three collections over the course of a semester resulting in a database of over 7100 images.

VI. Non-Orthogonal Iris Image Preprocessing

The preprocessing of iris images begins with the detection of the pupillary (inner) and limbic (outer) iris boundaries. In the case of orthogonal iris images, this involves locating sharp, circular edges that are relatively easy to find when the image is noiseless, meaning that the iris pixels are not hidden by glare, eyelids, and eyelashes. In the case of non-orthogonal iris images, the pupillary and limbic boundaries are now elliptical in shape rather than circular. This is a difficult problem to solve because there are a greater number of variable parameters with an ellipse than with a circle. In the case of circular iris boundaries, there are only two parameters, the radius of the circle and the location of the center of the circle. An ellipse has four variable parameters that must be determined. They include the lengths of the semi-major and semi-minor axes, the location of the center of the ellipse, and the amount of rotation of the ellipse. The primary objective of Ensign Bonney's 2004-2005 Trident Research was to devise an algorithm for detecting these elliptical boundaries and determining their

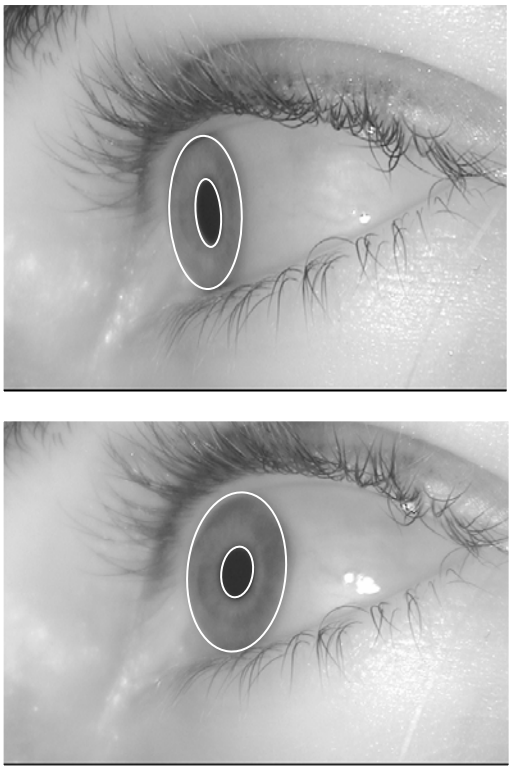

Figure 10. Detection of Elliptical Pupillary and Limbic Boundaries.

parameters (see Fig. 10) [1]. ENS Bonney's segmentation algorithm was used in the current research. This algorithm was the sole contribution of Ensign Bonney to the current Trident Project. The division of labor is delineated in Appendix A.

VII. Elliptical-to-Circular Coordinate Transformation

The second step in the preprocessing of an iris image is conversion of the image from rectangular to polar coordinates. However, in the case of a non-orthogonal iris image, this polar coordinate transformation no longer works and the iris template cannot be made because the iris is now effectively bounded by concentric ellipses rather than concentric circles. In order to rectify this situation, a function was written that performs an elliptical-to-circular coordinate transformation on the iris image. Then, the rectangular-to-polar coordinate transformation can still take place and the remainder of recognition can be performed. Ensign Bonney's non-orthogonal iris segmentation algorithm outputs all of the parameters needed to perform this transformation.

From the beginning, the assumption has been made that both the pupillary boundary ellipse and the limbic boundary ellipse have the same eccentricity, which is the ratio of their semi-major to semi-minor axis. This means that they are considered to be concentric ellipses even though their parameters are sometimes slightly different. Generally, they are close enough to being concentric, and the coordinate transformation still works well even if their parameters are not exactly the same. The general equation of an ellipse with semi-minor axis length a and semi-major axis length b (Fig. 11) is:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{1}$$

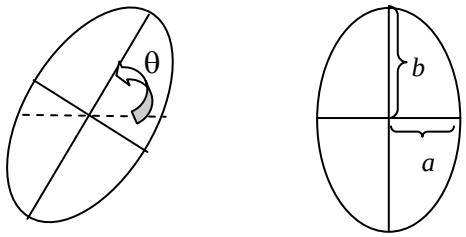

Figure 11. Ellipses with Rotation and Semi-Major and Semi-Minor Axes.

First, the angle of rotation of the ellipse is found, and the iris image is rotated by an angle (θ) of the same magnitude, but in the opposite direction so that the angle of rotation of the ellipse is now 0° (major axis is vertical). Then, in order to perform the elliptical-to-circular coordinate transformation, the *x*- and *y*-coordinate axes need to be redefined. The transformation function defines the new *x*′- and *y*′- coordinate axes to be:

$$x' = \frac{b}{a}x \qquad \text{and} \qquad y' = y \tag{2}$$

The *y*-axis is not changed in this transformation. The only change is that the *x*-axis is scaled by the ratio of *b* to *a*. In all of the collected non-orthogonal iris images, *b*, along the vertical axis, is longer than *a*, so this results in a "stretching out" of the horizontal axis. Substituting these equations into the standard ellipse equation results in (3).

$$\frac{1}{a^2}\left(\frac{a}{b}x'\right)^2 + \frac{y'^2}{b^2} = 1 \tag{3}$$

Making substitutions leads to (4), which has the equation of a circle with radius *b* in the new coordinate system.

$$x'^2 + y'^2 = b^2 \tag{4}$$

Figure 12 shows the iris preprocessing graphical user interface (GUI). The image is loaded, the segmentation operation is performed, and then the elliptical-to-polar transformation

function transforms the non-orthogonal iris to be circular in shape. The vertical thin black lines (meaning "no data") scattered throughout the image appear when the transformation takes place because when an elliptical iris image is made into a circle, it is missing some of the information in a fully circular, orthogonal iris image. This is because when the pixels go through the transformation in (2), the x'-coordinate values become more spaced out. For example, if the eccentricity is 2 (b=2, a=1), then x'=2x, and the vertical black lines would appear at every other column in the transformed image.

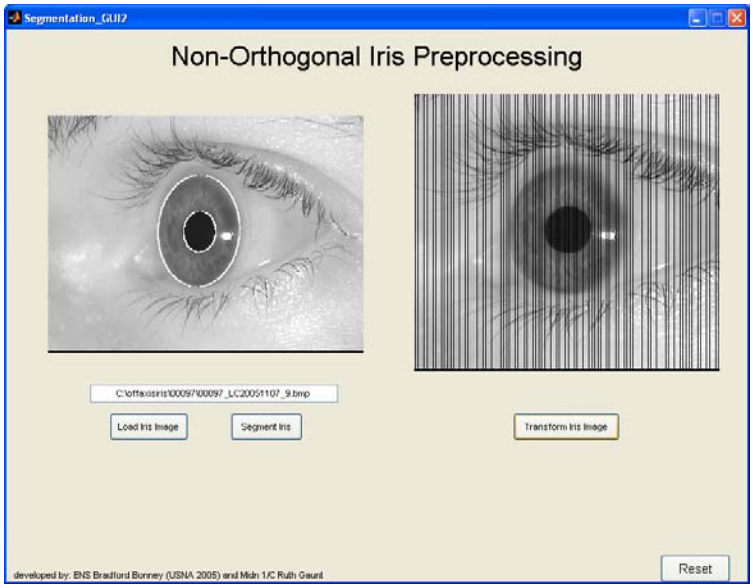

Figure 12. Preprocessing GUI.

Figure 13. Polar Transformation of Transformed Ellipse.

Since the pupillary and limbic boundaries of the iris are now approximately circular, the rectangular-to-polar coordinate transformation can now occur. The final result can be seen in Fig. 13. The black spots in the image are created when the black vertical lines ("no data") from the transformed image in Fig. 12 go through the unwrapping process. When the LTP image is produced, the LTP values will not be skewed by the black spots in the new image because these pixels will not be taken into consideration in the creation of the LTP. Ideally, the pupillary boundary should be a horizontal line, and further refinement and testing of the algorithm may

help to improve this.

VIII. Direct Ellipse Unwrapping

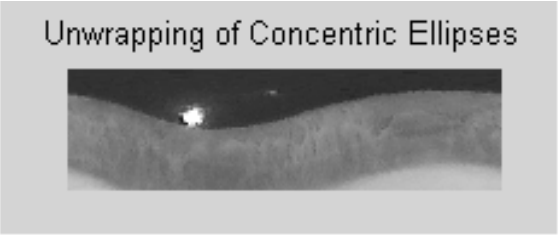

Figure 14. Direct Unwrapping of Concentric Ellipses.

Another method of non-orthogonal iris image preprocessing that has been experimented with is the direct unwrapping of ellipses without any elliptical-to-circular coordinate transformation. The algorithm starts with the limbic boundary and works inward toward the pupil, taking successively smaller concentric ellipses and then unwrapping them to create a row in the polar transformed image. As shown in Fig. 14, when the pupillary and limbic boundaries have close to the same centroids and are nearly concentric, the unwrapping process is successful and the pupillary and limbic boundaries are relatively equidistant.

On the other hand, if the centroids of the pupillary boundary ellipse and the limbic boundary ellipse are far apart, the direct ellipse unwrapping does not work. For example, if the algorithm is unwrapping ellipses based on the parameters of the limbic boundary ellipse, and if the pupillary boundary ellipse's centroid is very different, the pupil will actually not be unwrapped at all, and the result will be completely unusable (see Fig. 15).

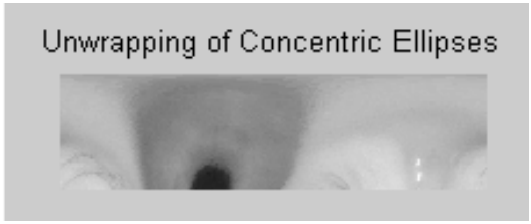

Figure 15. Poor Result for Direct Ellipse Unwrapping.

IX. Affine Transformation

The elliptical-to-circular coordinate transformation can also be performed by applying an affine transformation matrix to the image in MATLAB. This transformation matrix (5) scales the

horizontal axis of the image by ratio of semi-major axis to semi-minor axis as determined by the segmentation algorithm.

$$\begin{pmatrix} Ratio & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5}$$

The results of the affine transformation are displayed in Fig. 16. It is evident that the affine transformation outputs an image with more circular pupillary and limbic iris boundaries which can be detected by conventional orthogonal iris recognition algorithms. While this method may

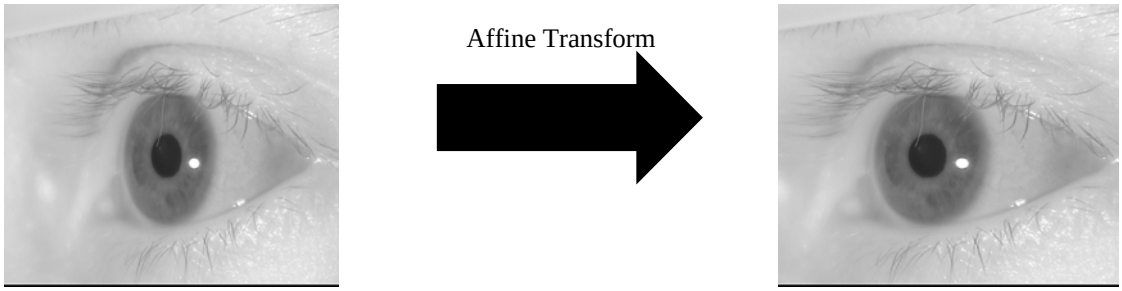

Figure 16. Affine Transformation.

seem preferable to the previously mentioned elliptical-to-circular transformation with the stripes of “no data,” it has one drawback that makes it less desirable for the non-orthogonal case. During this process, the transformation “smears” the iris pixels along the horizontal axis to create the circular iris boundaries. The smearing results in the interpolation of iris pixels values, which distorts the iris pixel information and could prevent accurate recognition. On the other hand, although the “no data” elliptical-to-circular transformation is more difficult to format with conventional iris recognition algorithms, this level of distortion is not present.

X. Modification of 1-D Algorithm

To test the feasibility of non-orthogonal iris recognition, the 1-D algorithm was modified to accommodate the transformed non-orthogonal images with the black, vertical “no data” lines. First, the algorithm was changed to take two input files: a bit map file of the transformed non-orthogonal iris image with the “no data” lines and a binary mask that has 1’s where the original image pixels are located and 0’s where the “no data” lines are. Both the image and the mask are then input into the preprocessing function. Because the transformed iris image now has circular pupillary and limbic boundaries, the algorithm used for pupillary boundary detection (edge detection and a Hough transform) can be applied to the image. When the rectangular-to-polar transformation occurs, the “no data” mask is used to prevent the “no data” pixels from being averaged with true iris pixels during the transformation process. Without the mask, the polar image of the iris would be distorted by the “no data” pixels, which would negatively impact the creation of iris templates to be used for recognition. Figures 17 and 18 show the process used to create an iris template from a non-orthogonal iris image.

When the segmentation algorithm fails to accurately locate the elliptical pupillary boundary, the ratio of semi-major axis to semi-minor axis of the pupillary boundary is also incorrect. This means that the pupillary and limbic boundaries of the transformed iris image are not circular, the iris cannot be accurately segmented, and iris template generation cannot occur. Figure 19 demonstrates an example of poor iris segmentation, incorrect elliptical-to-circular coordinate transformation, and failure to generate an iris template.

This failed result occurred because the pupillary boundary was improperly segmented, which in turn output a semi-major axis to semi-minor axis ratio that was much too high. This resulted in

an over-stretching of the image in the elliptical-to-circular transformation. In the transformed image, the pupillary boundary is now an ellipse with its major axis in the x-direction. This over-stretching also created an extremely dark transformed image because of the increased number of "no-data" pixels.

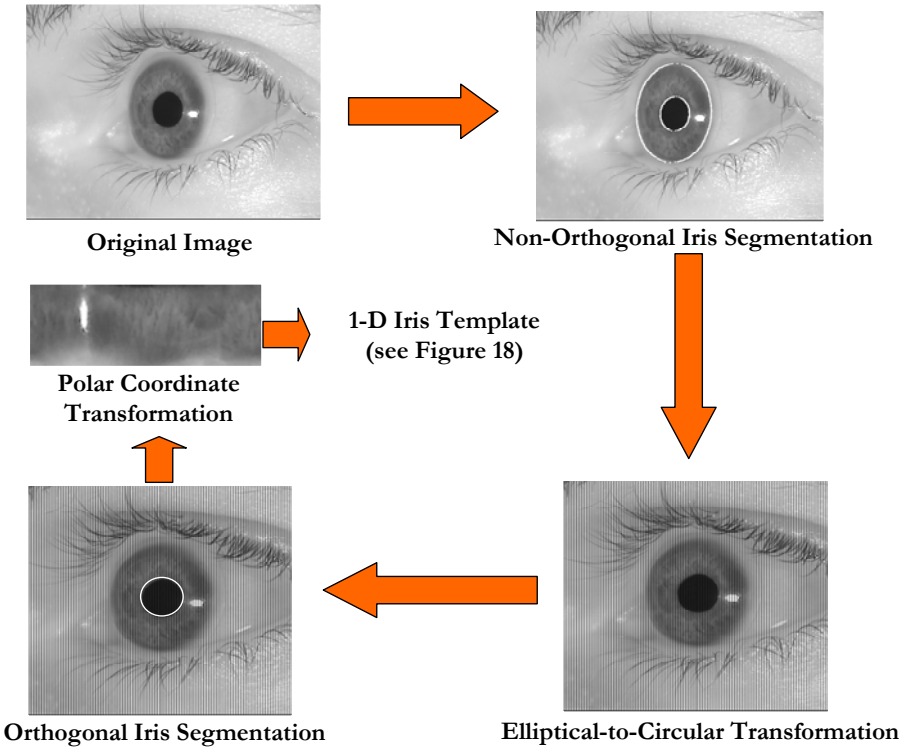

Figure 17. Non-Orthogonal Template Generation.

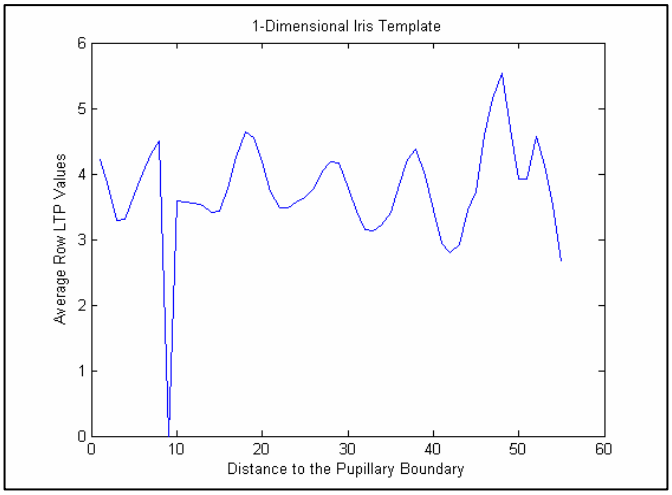

Figure 18. 1-D Iris Template.

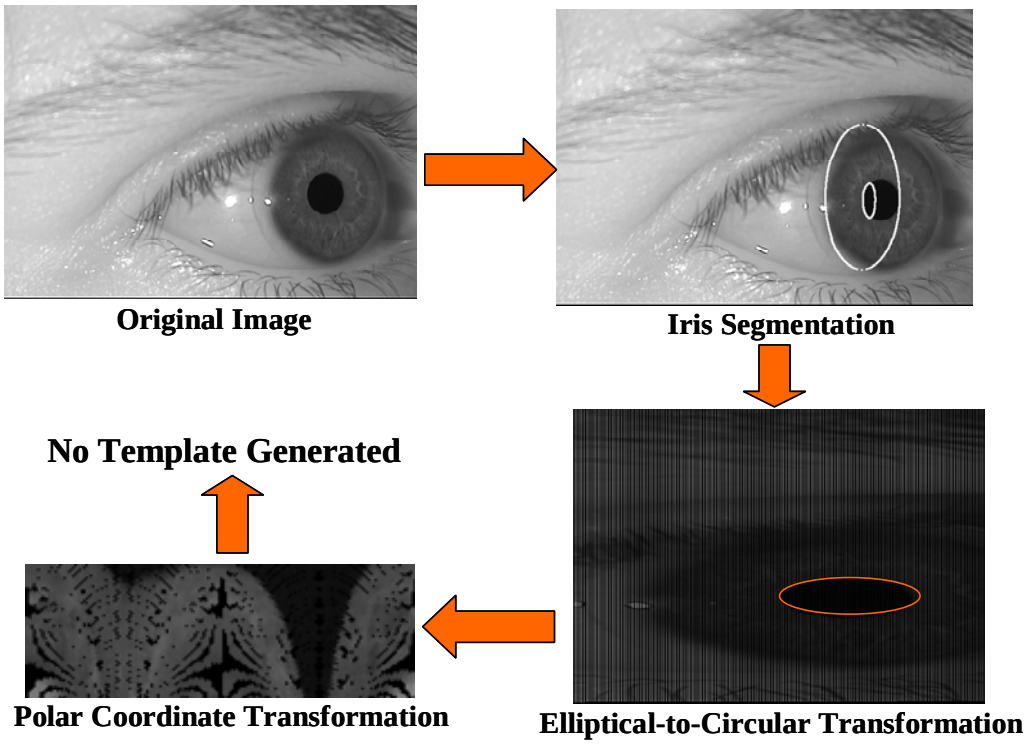

Figure 19. Failed Non-Orthogonal Template Generation.

XI. Determining Accuracy of Database Collection

After collection, the database images were run through the non-orthogonal iris segmentation algorithm that outputs the parameters of elliptical boundaries of the pupillary and limbic boundaries, including the centroid, semi-major axis length, and semi-minor axis length [4]. To assess the accuracy of collection at each non-orthogonal angle, the ratio of the semi-major axis length to semi-minor axis length of the pupillary boundary was calculated for each subject eye. Since the eccentricity of the pupillary boundary increases as the non-orthogonal imaging angle increases, the ratio of semi-major axis length to semi-minor axis length of the pupillary boundary should increase as well. Table 1 displays the mean ratio and standard deviation for each non-orthogonal angle. Images taken from an angle of zero degrees (orthogonal images) have the

smallest mean ratio of 1.085. This ratio is to be expected because the entire iris can be seen in the image, and in general, the limbic and pupillary boundaries of irises are approximately circular, which would translate to equal semi-major and semi-minor axes (ratio = 1.0). As the non-orthogonal imaging angle increases, the visible iris boundaries become more and more elliptical, and at an angle of 45°, the mean ratio was 1.3668.

Database Collection Analysis		
Angle	Mean Ratio	Standard Deviation
0°	1.085	0.3861
15°	1.1157	0.3750
30°	1.2147	0.3492
45°	1.3668	0.4227

Table 1. Iris data used for database analysis.

Figure 20 shows a histogram of the semi-major axis/semi-minor axis ratio values for images collected at the four different angles. This graph shows considerable overlap between the different non-orthogonal angles. In fact, the orthogonal and 15° degree images are virtually

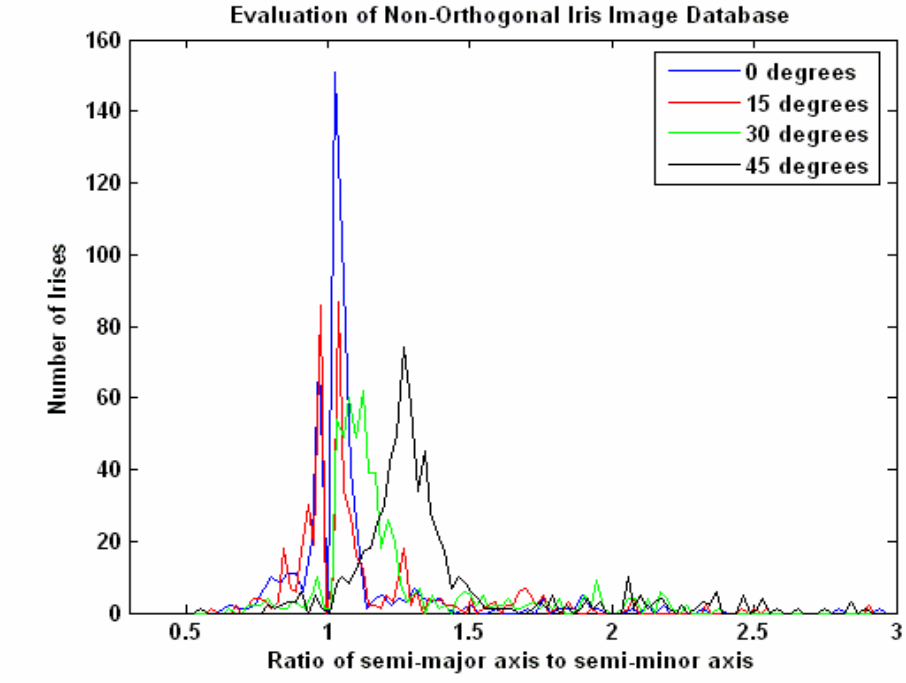

Figure 20. Analysis of non-orthogonal iris image database.

indistinguishable. Despite the overlap, the peak ratios for the 30° and 45° iris images are at increasingly higher ratios, which is expected. The variability among the histograms for each angle could be due to a few factors. One of these factors is that the person's head and chin are not completely restrained in the chin rest during collection. Another factor is that even though users have visual aids to fix their eyes on during collection, involuntary movement of the eye could cause variability in collection.

XII. Algorithm Performance

Figure 21 shows performance results using the 1-D algorithm on approximately 1250 orthogonal iris images collected as part of this project using a rank-matching curve. The horizontal axis shows the number ranked matches, and the vertical axis shows the percent accuracy. For any rank *n*, when presented with a new iris, the percent accuracy shows how often the correct iris was identified as being within the *n* closest irises from the database. As an example of how to read the curve, the correct eye was identified as one of the top ten (horizontal axis = 10) 76% of the time [5]. This curve for orthogonal recognition is being used as a baseline to measure the performance of non-orthogonal iris recognition.

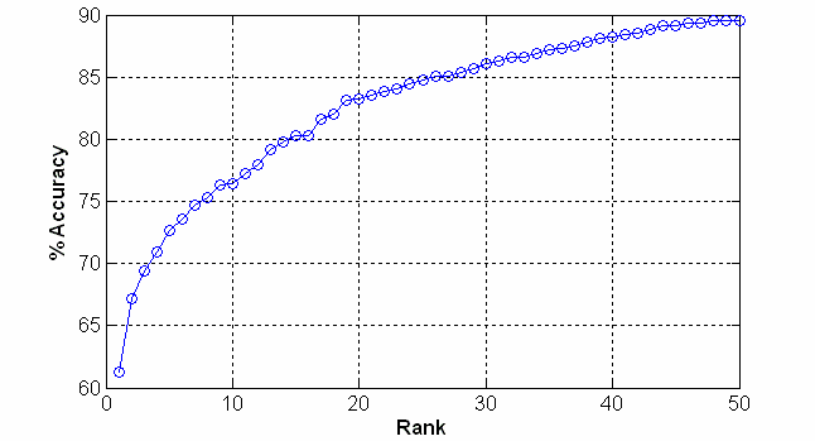

Figure 21. Rank-matching curve for orthogonal iris recognition.

Before testing was started, an enrollment template was created for each iris in the database (94 irises). An enrollment template consists of the average of four templates of the same iris. All of the images in the database were then compared to these enrollment templates, the Du measurements were calculated, and rank-matching curves were generated.

To test the feasibility of non-orthogonal iris recognition, five different sets of enrollment templates were created. First, an enrollment database of orthogonal templates was created by averaging four templates of each iris imaged at 0°. Similarly, three more enrollment databases were constructed from averaging four templates of iris images at each of the non-orthogonal angles (15°, 30°, and 45°). A final enrollment database was constructed from taking one template from each non-orthogonal angle for each iris and computing the average of the four templates.

Five tests were conducted by comparing templates of all images in the non-orthogonal database to each of the five enrollment databases. The process of iris segmentation, image transformation, template generation, and comparison takes about one minute for each image. The rank-matching curves are displayed in Figs. 22-25. From these graphs, it can be seen that the most accurate recognition occurs for iris images taken at the same non-orthogonal angle as the enrollment database. For instance, when being compared to an enrollment database of orthogonal images, over 60% of orthogonal images in the database were correctly ranked as the top match. Over 80% of the orthogonal images were correctly ranked in the top 20 matches. The rank-matching curve is lower than the “baseline” curve for orthogonal iris recognition because the orthogonal iris images went through the same elliptical-to-circular transformation process that the non-orthogonal images went through, and poor segmentation results would have distorted the orthogonal images. Similarly, for each of the other four tests, over 50% of the images in the

database that were captured at the same angle were correctly matched as the top-ranking iris, and over 80% were correctly ranked in the top 20.

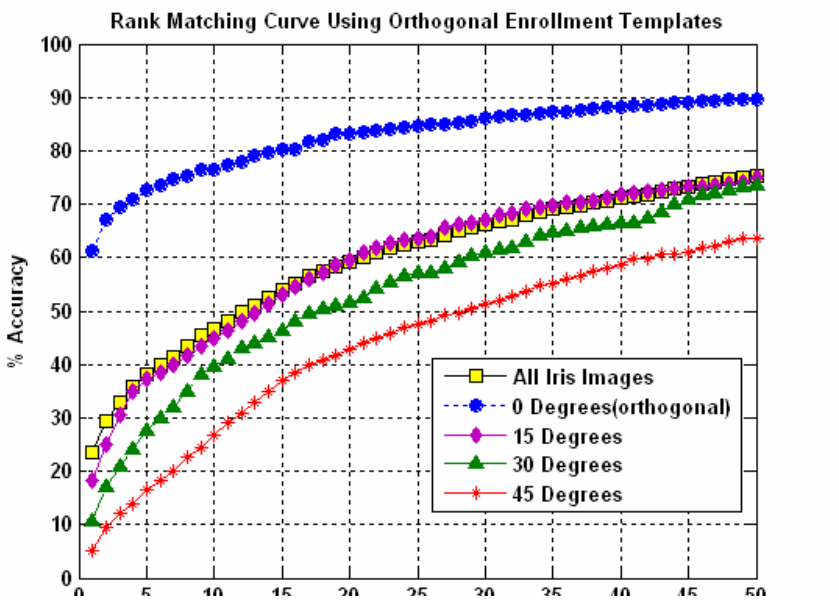

Figure 22. Rank-matching curve for orthogonal enrollment templates.

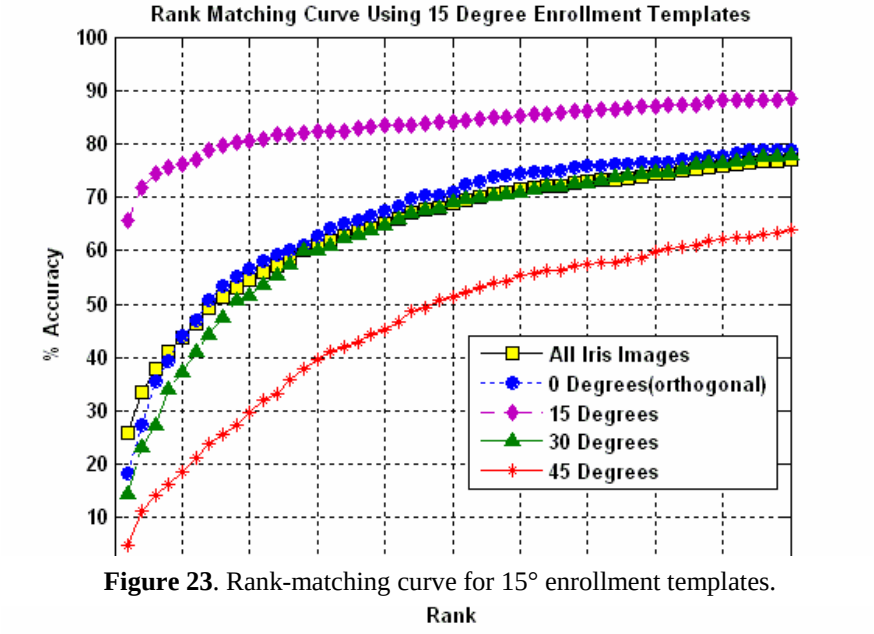

Figure 23. Rank-matching curve for 15° enrollment templates.

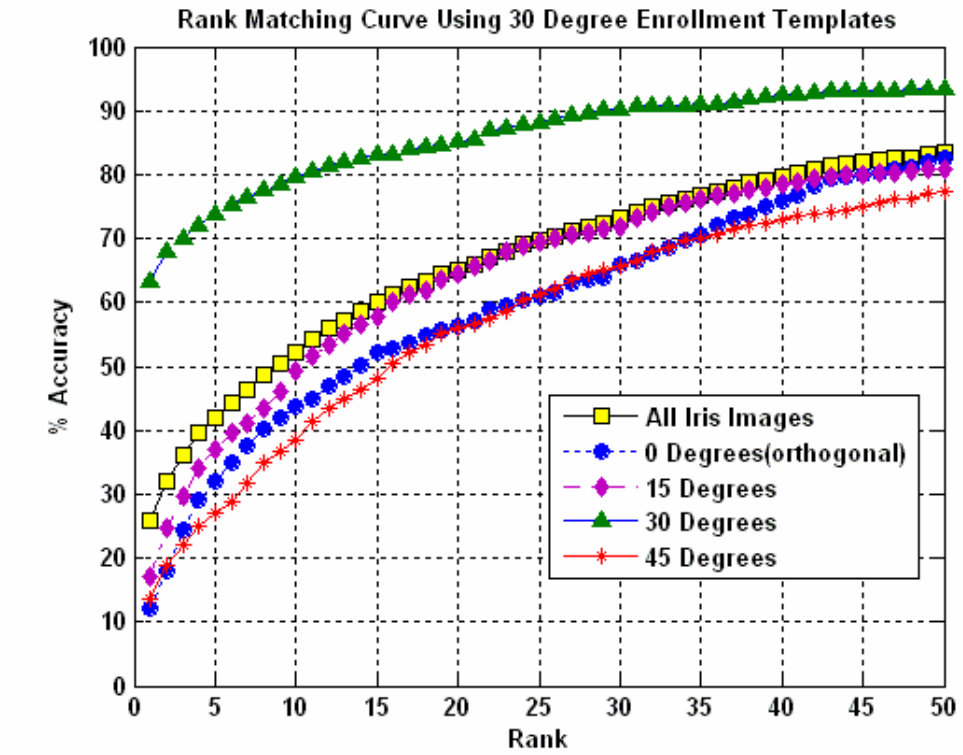

Figure 24. Rank-matching curve for 30° enrollment templates.

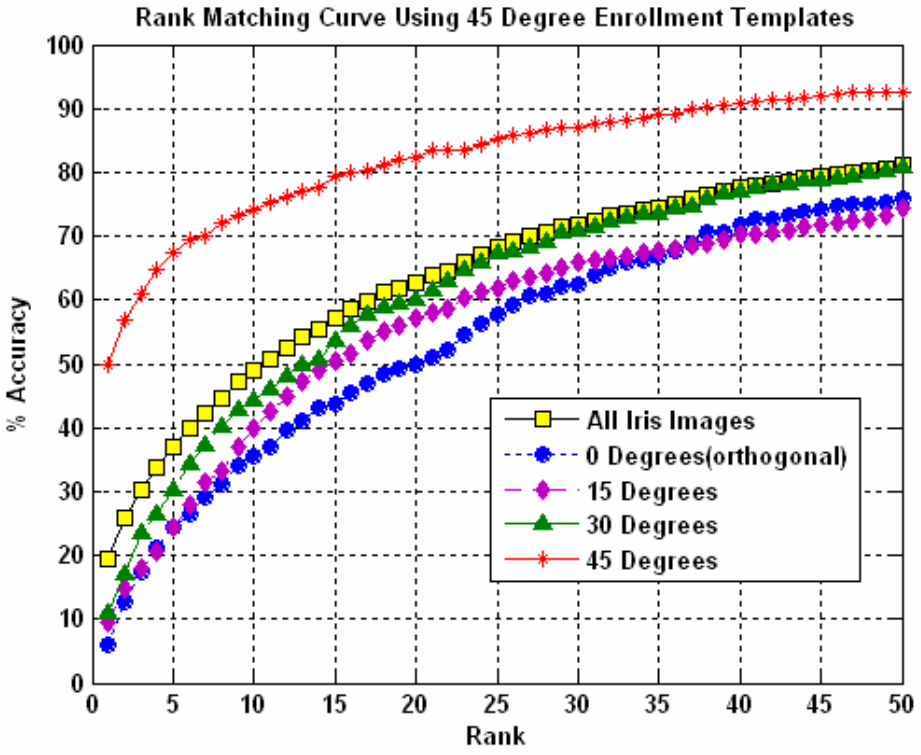

Figure 25. Rank-matching curve for 45° enrollment templates.

In addition, these rank-matching curves also show that the success of recognition is also dependent on the difference in angle between the enrollment template angle and the angle of the iris template to which it is being compared. For example, when being compared to an orthogonal enrollment template database, 15° iris images have better recognition results than 30° images, which in turn perform better than 45° images. The same is true for an enrollment database of 45° templates: 30° images have more accurate recognition results than 15°, which have better performance than orthogonal iris images.

The best rank matching curve for non-orthogonal iris recognition was produced when all the images in the database were compared to the database of enrollment templates that consist of an average of one template at each non-orthogonal angle (Fig. 26). The rank-matching curves for each non-orthogonal angle have better overall results for being ranked as the top match, and the

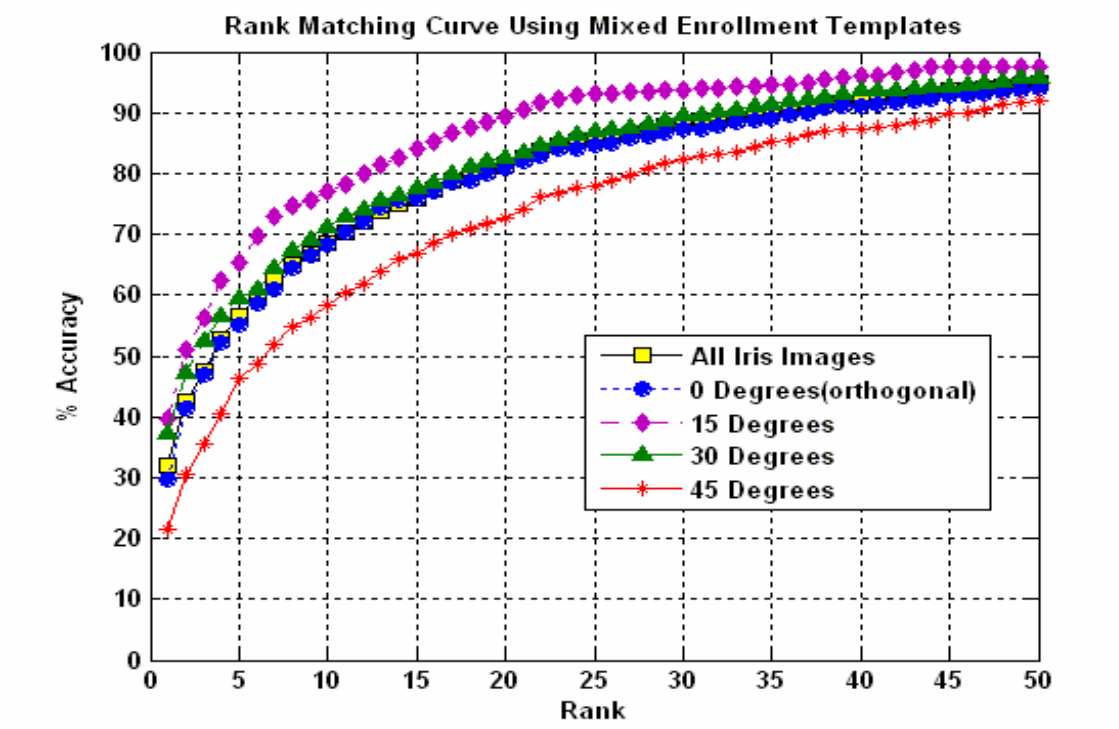

Figure 26. Rank-matching curve for mixed enrollment templates.

percent accuracy for being ranked in the top 20 irises is around 70% for all of the curves. The 15° and 30° images may have the best rank-matching curves because they are the two middle non-orthogonal angles. This makes sense because the enrollment template is an average of templates from each of the four non-orthogonal angles, and the average falls closer to the middle.

XIII. Conclusions

One of the difficulties with biometrics research is finding enough data, such as iris images, for testing. In the case of non-orthogonal iris recognition, there are presently only a few databases of non-orthogonal iris images, which make it difficult to develop robust recognition algorithms. This research has successfully produced a database of over 7100 images.

The variations in collection results displayed by the ratios of semi-major axis length to semi-minor axis length may have occurred for three reasons. First, even though the subject is instructed to stare straight ahead during the collection process and has visual aids at which to stare, there is no guarantee that the person was looking straight ahead at the instant the images were captured. In addition, the performance of the segmentation algorithm that finds the parameters of the elliptical iris boundaries is not perfect, and the true location of the boundaries is subjective. In addition, non-orthogonal iris images are not perfect ellipses because human iris shapes are not always perfectly circular, even in orthogonal images.

This research also resulted in the successful implementation of 1-D iris recognition for non-orthogonal iris images. The best results were produced when enrollment templates were made from averaging templates from each non-orthogonal angle and compared to all images in the non-orthogonal database. Also, for single-angle enrollment templates, the smaller the difference in angle between enrolled template and compared template, the better the results for recognition.

Accuracy of non-orthogonal iris recognition was lower than for orthogonal iris recognition for various reasons. The 1-D algorithm that was used in this research is not as accurate as other commercial algorithms because it discards a lot of information during iris preprocessing. Also, in cases of poor iris segmentation, the elliptical-to-circular transformation did not create circular pupillary and limbic boundaries, and iris templates could not properly be created. The elliptical-to-circular transformation itself was perhaps too simplistic a transformation to use because the iris is actually a three-dimensional structure, and the transformation function worked only in the x-y plane.

XIV. Future Work

The results of this research show the potential for the successful implementation of commercial non-orthogonal iris recognition algorithms and open the door to future research in this area. First, the elliptical-to-circular transformed images can be formatted to work with other orthogonal iris recognition algorithms to see if better results are achieved than with the 1-D algorithm. In addition, the elliptical-to-circular transformation should be refined so that all three dimensions are considered. In order for this to work, the elliptical iris boundaries also need to be more accurately segmented.

More images can also be added to the non-orthogonal iris image database, and more refined methods for determining the accuracy of angular capture can be created. First, using three-dimensional rotation and projection, expected values for ratios of semi-major axis to semi-minor axis could be found for each non-orthogonal angle. That way, the ratio of each incoming image can be compared to the expected ratio.

XV. Works Cited

[1] B. Bonney, “Non-Orthogonal Iris Localization,” *Final U.S. Naval Academy Trident Report* Apr. 2005.

[2] Y. Du, R.W. Ives, D.M. Etter, T.B. Welch, and C.-I Chang, "One Dimensional Approach to Iris Recognition", *Proceedings of the SPIE*, pp. 237-247, Apr., 2004.

[3] Y. Du, R.W. Ives, and D.M. Etter, "Iris Recognition", The Electrical Engineering Handbook, 3rd Edition, Boca Raton, FL: CRC Press, 2006, pp. 25-1 to 25-10.

[4] B.L. Bonney, R.W. Ives, D.M. Etter and Y. Du, "Iris Pattern Extraction using Bit-Planes and Standard Deviations," Proc. of the 38th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, pp. 582-586, Nov. 2004.

[5] Y. Du, R.W. Ives, D.M. Etter, and T.B. Welch, "Use of One-Dimensional Iris Signatures to Rank Iris Pattern Similarities," Optical Engineering, 2006 (in press).

[6] R. Schultz and R.W. Ives, "Biometric Data Acquisition using MATLAB GUIs," Proceedings of the 2005 IEEE Frontiers in Education Conference, Indianapolis, IN, October 2005, pp. S1G-1 to S1G-5.

[7] Y. Du, B. Bonney, R.W. Ives, D.M. Etter, and R. Schultz, "Analysis of Partial Iris Recognition using a 1D Approach," Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Vol. II, pp. 961-964, Mar. 2005.

XVI. Works Consulted

1. Calvert, J.B. “Ellipse,” *Dr. James B. Calvert.* 31 May 2005.
<http://www.du.edu/~jcalvert/math/ellipse.htm>.

2. Daugman, John. *How Iris Recognition Works.* 25 Oct 2003. <www.cl.cam.ac.uk/users/jgd1000/irisrecog.pdf>.

3. Y. Du, B. Bonney, R.W. Ives, D.M. Etter, and R. Schultz, "Analysis of Partial Iris Recognition using a 1D Approach," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP), Vol. II, pp. 961-964, Mar. 2005.

4. Weisstein, Eric W. "Ellipse." From *MathWorld*--A Wolfram Web Resource. 24 May 2005. <http://mathworld.wolfram.com/Ellipse.html>

XVII. Appendices

Appendix A: Division of Work

Appendix B: MATLAB code written for non-orthogonal iris preprocessing.

Appendix C: Experimental Data

Appendix D: Publications

Appendix A: Division of Work

- **Ensign Bonney's Contribution:**
    - Developed algorithm for detecting elliptical iris boundaries
    - Determined parameters of ellipses for segmentation
    - Developed a GUI to display segmentation results
- **MIDN 1/C Gaunt's Fall 2005 Semester Contribution:**
    - Construction of Non-Orthogonal Iris Collection Station
        - Collection of data for 40 irises
    - Development of two methods for non-orthogonal iris image preprocessing
        - Elliptical-to-circular coordinate transformation
        - Direct unwrapping of concentric ellipses
        - Modified segmentation GUI to display elliptical-to-circular transformation
- **MIDN 1/C Gaunt's Spring 2006 Semester Contribution:**
    - Modification of 1-D algorithm to accommodate use of non-orthogonal iris images
    - Testing of non-orthogonal iris recognition algorithm
    - Ongoing collection of irises
    - Submission and presentation to the 2006 National Conference for Undergraduate Research

Appendix B: MATLAB Code

Function List:

```
function [v,w,h_cent,g_cent,max_radius_l,max_radius_p,ratio] =
transform_iris(p_stats,i_stats,iris)

%This function takes the parameters for the pupillary (inner) and limbic
%(outer) boundaries of the iris and performs the elliptical-to-circular
%coordinate transformtion.
%
%  usage: [v,h_cent,g_cent,max_radius]=transform_iris(p_stats,i_stats,iris)
%
%where v is the transformed image, h_cent and g_cent are the x- and
%y-coordinates of the transformed circle,  max_radius is the radius of
%the transformed circle, p_stats and i_stats are structures that contain
%the parameters of the pupillary and limbic ellipses, and iris is the
%original iris image.
%
%Author: MIDN 1/C Ruth Gaunt

angle = p_stats.Orientation

i_semi_x = round(i_stats.MinorAxisLength/2);
i_semi_y = round(i_stats.MajorAxisLength/2);
i_cent_x = round(i_stats.Centroid(1));
i_cent_y = round(i_stats.Centroid(2));

p_semi_x = round(p_stats.MinorAxisLength/2);
p_semi_y = round(p_stats.MajorAxisLength/2);
p_cent_x = round(p_stats.Centroid(1));
p_cent_y = round(p_stats.Centroid(2));

%Rotates iris image so that the orientation is 90 degrees, meaning that the
%rotation parameter of the ellipse is eliminated.

if abs(p_semi_y - p_semi_x)<=10
   iris_new = iris;
else
   if angle>0
      iris_new = imrotate(iris,(90-angle),'nearest','crop');
   elseif angle<0
      iris_new = imrotate(iris,-(90+angle),'nearest','crop');
   elseif angle == 0
      iris_new = imrotate(iris,0,'nearest','crop');
   end
end
```

```
bitplane_zero2 = adjusted_bitzero(iris_new);

[pupil_mask2, stats2] = pupil_morph2(bitplane_zero2);

%figure(2), imshow(iris_new)



k_init = i_cent_y - i_semi_y-100
if k_init < 1
   k_init = 1;
end
k_final = i_cent_y + i_semi_y+100
if k_final>480
   k_final=480;
end

l_init = i_cent_x - i_semi_x-100
if l_init < 1
   l_init = 1;
end

l_final = i_cent_x + i_semi_x+100
if l_final > 640
   l_final = 640;
end

dist_major = i_semi_y - p_semi_y
dist_minor = i_semi_x - p_semi_x


%Performs the elliptical-to-circular coordinate transformation
for k = 1:480
   for l = l_init: l_final
      x_init = l_init - i_cent_x;
      x = l - i_cent_x;
      m_init = round((p_semi_x/p_semi_y)*x_init) + p_cent_x;
      m = round((p_semi_x/p_semi_y)*x)+ p_cent_x;
      g = k - k_init +1;
      h = m - m_init +1;
      q(k,h) = iris_new(k,l);
      b(k,h) = 1;
   end
end
```

```
h_cent = p_cent_x - m_init + 1;
g_cent = p_cent_y - k_init + 1;
max_radius_l = i_semi_y;
max_radius_p = p_semi_y;
[s,a]= size(q);

h_init = h_cent-319;
h_final = h_cent+320;

% x_final = l_final - i_cent_x;
% m_final = round((i_semi_x/i_semi_y)*x_final)+ i_cent_x;

if h_init<=0
    h_init = 1;
end

if a>640
   v = imcrop(q,[(h_init) 1 639 479]);
   w = imcrop(b,[(h_init) 1 639 479]);
else
    v = q;
    w = b;
end
size(v)

figure(2), imshow(v)
figure(6), imshow(w)
% imwrite(uint8(v),'iris_transform.bmp')
% imwrite(uint8(w),'iris_transform_mask.bmp')
```

```
function varargout = Segmentation_GUI2(varargin)
% SEGMENTATION_GUI2 M-file for Segmentation_GUI2.fig
%      SEGMENTATION_GUI2, by itself, creates a new SEGMENTATION_GUI2 or raises the
existing
%      singleton*.
%
%      H = SEGMENTATION_GUI2 returns the handle to a new SEGMENTATION_GUI2 or
the handle to
%      the existing singleton*.
%
%      SEGMENTATION_GUI2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SEGMENTATION_GUI2.M with the given input
arguments.
%
%      SEGMENTATION_GUI2('Property','Value',...) creates a new SEGMENTATION_GUI2 or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Segmentation_GUI2_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Segmentation_GUI2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Copyright 2002-2003 The MathWorks, Inc.

% Edit the above text to modify the response to help Segmentation_GUI2

% Last Modified by GUIDE v2.5 21-Nov-2005 22:22:49

% Begin initialization code - DO NOT EDIT
% Authors: ENS B. Bonney (USNA 2005), MIDN 1/C R. Gaunt
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Segmentation_GUI2_OpeningFcn, ...
                   'gui_OutputFcn',  @Segmentation_GUI2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Segmentation_GUI2 is made visible.
function Segmentation_GUI2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Segmentation_GUI2 (see VARARGIN)

% Choose default command line output for Segmentation_GUI2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Segmentation_GUI2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

%Turn off initial axes1 and axes2 for GUI execution
set(handles.axes1, 'HandleVisibility', 'ON');
axes(handles.axes1);
axis off;
title(' ');
set(handles.axes1, 'HandleVisibility', 'OFF');

set(handles.axes2, 'HandleVisibility', 'ON');
axes(handles.axes2);
axis off;
title(' ');
set(handles.axes2, 'HandleVisibility', 'OFF');
set(handles.text20, 'String', ' ');
set(handles.text21, 'String', ' ');


mex localthresh.c
% --- Outputs from this function are returned to the command line.
```

```
function varargout = Segmentation_GUI2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in transform.
function loadraw_Callback(hObject, eventdata, handles)
% hObject    handle to transform (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.text4, 'String', ' ');
set(handles.numtruepixel, 'String', ' ');
set(handles.text6, 'String', ' ');
set(handles.nummaskpixel, 'String', ' ');
set(handles.text10, 'String', ' ');
set(handles.lowqual, 'String', ' ');
set(handles.text12, 'String', ' ');
set(handles.upperqual, 'String', ' ');
set(handles.text15, 'String', ' ');
set(handles.text18, 'String', ' ');
set(handles.topqual, 'String', ' ');
set(handles.text16, 'String', ' ');
set(handles.numcommonpixel, 'String', ' ');

global iris_image;
filename = get(handles.filename, 'String');

iris_image = imread(filename);

set(handles.axes1, 'HandleVisibility', 'ON');
axes(handles.axes1);
%gimage(norim(iris_image)), axis image;
imshow(iris_image,[])
axis off;
set(handles.axes1, 'HandleVisibility', 'OFF');

global truth_mask;
global iris_image;
```

```
truth_mask = get_mask(iris_image);


% --- Executes on button press in segment.
function segment_Callback(hObject, eventdata, handles)
% hObject    handle to segment (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global iris_image2;
global iris_mask;
global stats;


set(handles.text20, 'String', ' ');
set(handles.text21, 'String', ' ');


set(handles.text20, 'String', 'Segmenting...');
pause(0.01);

[iris_mask, p_stats, i_stats] = iris_segmentation(iris_image2);
save eye.mat p_stats i_stats;

set(handles.axes2, 'HandleVisibility', 'ON');
axes(handles.axes2);
temp = uint8(bwmorph(iris_mask, 'dilate', 1));
%gimage(norim(norim(uint8(temp)) + iris_image2)), axis image;
temp = logical(temp);
iris_image3 = iris_image2;
iris_image3(temp)=255;
imshow(iris_image3,[])
set(handles.axes2, 'HandleVisibility', 'OFF');

set(handles.text20, 'String', ' ');

% --- Executes on button press in loadtruth.
function transform_Callback(hObject, eventdata, handles)
% hObject    handle to loadtruth (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global iris_image2;
global iris_mask;
```

```
global stats;


set(handles.text20, 'String', ' ');
set(handles.text21, 'String', ' ');


set(handles.text21, 'String', 'Transforming...');
pause(0.01);

%[iris_mask, p_stats, i_stats] = iris_segmentation(iris_image2);
load eye.mat;
[v, h_cent, g_cent, max_radius] = transform_iris(p_stats,i_stats,iris_image2);

set(handles.axes1, 'HandleVisibility', 'ON');
axes(handles.axes1);
%gimage(v), axis image;
imshow(v, [])
axis off;
set(handles.axes1, 'HandleVisibility','OFF');
set(handles.text21,'String',' ');


% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes1


% --- Executes during object creation, after setting all properties.
function axes2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes2


% --- Executes during object creation, after setting all properties.
function transform_CreateFcn(hObject, eventdata, handles)
% hObject    handle to transform (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes during object creation, after setting all properties.
function loadtruth_CreateFcn(hObject, eventdata, handles)
% hObject    handle to loadtruth (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes during object creation, after setting all properties.
function savetruth_CreateFcn(hObject, eventdata, handles)
% hObject    handle to savetruth (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes during object creation, after setting all properties.
function segment_CreateFcn(hObject, eventdata, handles)
% hObject    handle to segment (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes on button press in Reset.
function Reset_Callback(hObject, eventdata, handles)
% hObject    handle to Reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.axes1, 'HandleVisibility', 'ON');
axes(handles.axes1);
cla reset;
axis off;
title(' ');
set(handles.axes1, 'HandleVisibility', 'OFF');

set(handles.axes2, 'HandleVisibility', 'ON');
axes(handles.axes2);
cla reset;
axis off;
title(' ');
set(handles.axes2, 'HandleVisibility', 'OFF');
```

```
set(handles.text4, 'String', ' ');
set(handles.numtruepixel, 'String', ' ');
set(handles.text6, 'String', ' ');
set(handles.nummaskpixel, 'String', ' ');
set(handles.text10, 'String', ' ');
set(handles.lowqual, 'String', ' ');
set(handles.text12, 'String', ' ');
set(handles.upperqual, 'String', ' ');
set(handles.text15, 'String', ' ');
set(handles.text18, 'String', ' ');
set(handles.topqual, 'String', ' ');
set(handles.text16, 'String', ' ');
set(handles.numcommonpixel, 'String', ' ');


% --- Executes during object creation, after setting all properties.
function Reset_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called




function filename_Callback(hObject, eventdata, handles)
% hObject    handle to filename (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of filename as text
%        str2double(get(hObject,'String')) returns contents of filename as a double


% --- Executes during object creation, after setting all properties.
function filename_CreateFcn(hObject, eventdata, handles)
% hObject    handle to filename (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
```

```
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on button press in calculate.
function calculate_Callback(hObject, eventdata, handles)
% hObject    handle to calculate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global iris_image;
global iris_image2;
global truth_mask;
global iris_mask;
global stats;

%fill mask
n = 4;
temp_mask = ones(480, 640);
while sum(~temp_mask(:)) == 0
   temp_mask = bwmorph(iris_mask, 'dilate', n);
   temp_mask = imfill(temp_mask, [1 1]);
   location = round([stats.Centroid(2) stats.Centroid(1)]);
   temp_mask = imfill(temp_mask, location);
   n = n + 1;
end

iris_mask = ~temp_mask;
iris_mask = bwmorph(iris_mask, 'dilate', n-2);

set(handles.axes2, 'HandleVisibility', 'ON');
axes(handles.axes2);
%gimage(norim(norim(uint8(iris_mask)) + iris_image2)), axis image;
imshow(uint8(iris_mask) + iris_image2,[])
axis off;
set(handles.axes2, 'HandleVisibility', 'OFF');

%%
combo = truth_mask & iris_mask;
num_common_pixels = sum(combo(:));
num_true_pixels = sum(truth_mask(:));
num_mask_pixels = sum(iris_mask(:));
```

```
num_error_pixels = num_mask_pixels - num_common_pixels;
if num_error_pixels < 0
   num_error_pixels = 0;
end

low = (num_common_pixels - 0.1 * num_error_pixels) / num_true_pixels;
mid = (num_common_pixels - 0.4 * num_error_pixels) / num_true_pixels;;
top = (num_common_pixels - 0.7 * num_error_pixels) / num_true_pixels;;

set(handles.text4, 'String', 'Number of True Iris Pixels:');
set(handles.numtruepixel, 'String', num_true_pixels);
set(handles.text6, 'String', 'Number of Mask Iris Pixels:');
set(handles.nummaskpixel, 'String', num_mask_pixels);
set(handles.text10, 'String', '10% Quality Bound:');
set(handles.lowqual, 'String', low);
set(handles.text12, 'String', '40% Quality Bound:');
set(handles.upperqual, 'String', mid);
set(handles.text18, 'String', '70% Quality Bound:');
set(handles.topqual, 'String', top);
set(handles.text16, 'String', 'Number of Common Pixels:');
set(handles.numcommonpixel, 'String', num_common_pixels);




% --- Executes during object creation, after setting all properties.
function calculate_CreateFcn(hObject, eventdata, handles)
% hObject    handle to calculate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes during object creation, after setting all properties.
function text15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called




% --- Executes on button press in loadiris2.
function loadiris2_Callback(hObject, eventdata, handles)
% hObject    handle to loadiris2 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.text20, 'String', ' ');
set(handles.text21, 'String', ' ');


global iris_image2;

filename2 = get(handles.edit2, 'String')

iris_image2 = imread(filename2);

set(handles.axes2, 'HandleVisibility', 'ON');
axes(handles.axes2);
%gimage(norim(iris_image2)), axis image;
imshow(iris_image2,[])
axis off;
set(handles.axes2, 'HandleVisibility', 'OFF');

% --- Executes during object creation, after setting all properties.
function loadiris2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to loadiris2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called




function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



% --- Executes during object creation, after setting all properties.
function numcommonpixel_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numcommonpixel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes during object creation, after setting all properties.
function text16_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
function varargout = iriscapture(varargin)
%  Offaxis iriscapture v .0
%
%
% IRISCAPTURE M-file for iriscapture.fig  version 0.2
%      IRISCAPTURE, by itself, creates a new IRISCAPTURE or raises the existing
%      singleton*.
%
%      H = IRISCAPTURE returns the handle to a new IRISCAPTURE or the handle to
%      the existing singleton*.
%
%      IRISCAPTURE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in IRISCAPTURE.M with the given input arguments.
%
%      IRISCAPTURE('Property','Value',...) creates a new IRISCAPTURE or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before iriscapture_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to iriscapture_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
global IRIS_ROOT;
global IRIS_DB;
global FRAMES_PER_TRIGGER
global PERIOD;
global ACTUAL_FRAME_RATE
global DESIRED_FRAME_RATE

IRIS_ROOT='c:\offaxisiris';
IRIS_DB='\irisdb.csv';
PERIOD=1;
ACTUAL_FRAME_RATE=30;  % 30 frames per second
DESIRED_FRAME_RATE=10; % 10 frames per second
FRAMES_PER_TRIGGER=9;

% Edit the above text to modify the response to help iriscapture

% Last Modified by GUIDE v2.5 16-Oct-2005 16:59:45

% Begin initialization code - DO NOT EDIT
% Authors: R.C. Schultz, MIDN 1/C R.M. Gaunt
```

```
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @iriscapture_OpeningFcn, ...
                   'gui_OutputFcn',  @iriscapture_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

camera_timer = timer('TimerFcn',@Timer_Call,'Period', 45.0, 'ExecutionMode','fixedDelay');
start(camera_timer);


function Timer_Call(handle, obj)
a = serial('COM1');
fopen(a);
c = '197 240 114';
fprintf(a,c);
fclose(a);

% --- Executes just before iriscapture is made visible.
function iriscapture_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to iriscapture (see VARARGIN)

% Choose default command line output for iriscapture
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes iriscapture wait for user response (see UIRESUME)
```

```
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = iriscapture_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double

% --- Executes on button press in Obstructed.
function Obstructed_Callback(hObject, eventdata, handles)
% hObject    handle to Obstructed (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of Obstructed


% --- Executes on button press in Trauma.
function Trauma_Callback(hObject, eventdata, handles)
% hObject    handle to Trauma (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Trauma


% --- Executes on button press in Surgery.
function Surgery_Callback(hObject, eventdata, handles)
% hObject    handle to Surgery (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Surgery


% --- Executes on button press in checkbox4.
function checkbox4_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox4


% --- Executes on button press in Glasses.
function Glas_Callback(hObject, eventdata, handles)
% hObject    handle to Glasses (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Glasses


% --- Executes on button press in checkbox6.
function checkbox6_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of checkbox6


% --- Executes on button press in lefteye.
function lefteye_Callback(hObject, eventdata, handles)
% hObject    handle to lefteye (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of lefteye


% --- Executes on button press in righteye.
function righteye_Callback(hObject, eventdata, handles)
% hObject    handle to righteye (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of righteye


% --- Executes on button press in Female.
function radiobutton3_Callback(hObject, eventdata, handles)
% hObject    handle to Female (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Female


% --- Executes on button press in Female.
function Female_Callback(hObject, eventdata, handles)
% hObject    handle to Female (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Female

% --- Executes on button press in angle0.
function angle0_Callback(hObject, eventdata, handles)
% hObject    handle to angle0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of angle0

% --- Executes on button press in angle15.
function angle15_Callback(hObject, eventdata, handles)
% hObject    handle to angle0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of angle15

% --- Executes on button press in angle30.
function angle30_Callback(hObject, eventdata, handles)
% hObject    handle to angle0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of angle30

% --- Executes on button press in angle45.
function angle45_Callback(hObject, eventdata, handles)
% hObject    handle to angle0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of angle45

% --- Executes during object creation, after setting all properties.
function IrisAge_CreateFcn(hObject, eventdata, handles)
% hObject    handle to IrisAge (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function IrisAge_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to IrisAge (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of IrisAge as text
%        str2double(get(hObject,'String')) returns contents of IrisAge as a double


% --- Executes during object creation, after setting all properties.
function IrisColor_CreateFcn(hObject, eventdata, handles)
% hObject    handle to IrisColor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on selection change in IrisColor.
function IrisColor_Callback(hObject, eventdata, handles)
% hObject    handle to IrisColor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns IrisColor contents as cell array
%        contents{get(hObject,'Value')} returns selected item from IrisColor


% --- Executes on button press in previewbutton.
function previewbutton_Callback(hObject, eventdata, handles)
% hObject    handle to previewbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of previewbutton

function filename = GetFilename( handles )
% Generate and return the filename based on parameters
global IRIS_ROOT;
```

```
global IRIS_DB;
directory = GetDirectory( handles );
Subject=GetSubjectNumber(handles);
Eye=GetEye(handles);
Number=get(handles.ImageNumber,'String');
Normal=GetGlassesorContacts(handles); % wearing glasses, contacts or normal
filename=sprintf('%s\\%s_%s%s%s_%s.bmp',directory,Subject,Eye,Normal,datestr(now,'yyyym
mdd'),Number);

function directory = GetDirectory( handles )
% Return the directory for the current Subject
global IRIS_ROOT;
global IRIS_DB;
Subject=GetSubjectNumber(handles);
directory=sprintf('%s\\%s',IRIS_ROOT,Subject);

function normal = GetGlassesorContacts( handles )
% Return N - no glasses
% return G - Glasses
% return C - Contacts
if ( get(handles.Glasses,'Value') ),
   normal='G';
else
   if ( get(handles.Contacts,'Value') ),
      normal = 'C';
   else
      normal = 'N';
   end
end

% --- Executes on button press in SaveImage.
function SaveImage_Callback(hObject, eventdata, handles)
% hObject    handle to SaveImage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global IRIS_ROOT;
global IRIS_DB;
global g_frame;
Subject=GetSubjectNumber(handles);
Eye=GetEye(handles);
Sex=GetSex(handles);
Age=GetIrisAge(handles);
Color=GetIrisColor(handles);
Angle=GetIrisAngle(handles);
```

```
Details=GetSubjectDetails(handles);

directory=GetDirectory( handles );
filename=GetFilename( handles );
if ( exist(filename) == 2 )
   button=questdlg('This file exists! \nAre you sure want to overwrite it?');
else
   button = 'Yes';
end
if ( button == 'Yes')

   if ( exist(directory) == 7 )
      imwrite(g_frame,filename,'bmp');
   else
      mkdir(directory);
      imwrite(g_frame,filename,'bmp');
   end
   fid=fopen(sprintf('%s%s',IRIS_ROOT,IRIS_DB),'a');

imageinfo=sprintf('%s,%s,%s,%s,%s,%s,%s,%s',Subject,Eye,Sex,Age,Color,Details,Angle,filen
ame);
   fprintf(fid,'%s\n',imageinfo);
   fclose(fid);
   set(handles.FileName,'String',imageinfo);
end

% --- Executes during object creation, after setting all properties.
function ImageNumber_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ImageNumber (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
set(hObject,'String','1');
set(hObject,'Value',1);
```

```
function ImageNumber_Callback(hObject, eventdata, handles)
% hObject    handle to ImageNumber (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ImageNumber as text
%        str2double(get(hObject,'String')) returns contents of ImageNumber as a double
curr=get(hObject,'String');
set(hObject,'Value',str2num(curr));


% --- Executes during object creation, after setting all properties.
function DeviceList_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DeviceList (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
DeviceList_LoadListBox(hObject)

function DeviceList_LoadListBox(hObject)
% hObject   handle to DeviceList
global g_vidObjects;
global g_numdrivers;
global g_drivers;
hwinfo = imaqhwinfo;
g_drivers = hwinfo.InstalledAdaptors;
g_numdrivers = max(size( g_drivers ));
set(hObject,'String',g_drivers);
g_vidObjects=videoinput(char(g_drivers(1)),1)
if g_vidObjects.Name=='M_RS170-matrox-1',
   set(g_vidObjects,'SelectedSourceName','CH2');
end
%if g_vidObjects
for a = 2:g_numdrivers-1,
  g_vidObjects(a)=videoinput(char(g_drivers(a)),a);
end
```

```
% --- Executes on selection change in DeviceList.
function DeviceList_Callback(hObject, eventdata, handles)
% hObject    handle to DeviceList (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns DeviceList contents as cell array
%        contents{get(hObject,'Value')} returns selected item from DeviceList


% --- Executes on button press in CaptureImage.
function CaptureImage_Callback(hObject, eventdata, handles)
% hObject    handle to CaptureImage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global g_vidObjects
global g_drivers;
global g_frame;
DeviceList=get(handles.DeviceList);
vidObj=g_vidObjects(DeviceList.Value);
g_frame=getsnapshot(vidObj);
%figure(handles.figure1);
image(g_frame);
tmp=g_drivers(DeviceList.Value)
if size(char(tmp)) == size('matrox')
   if char(tmp) == 'matrox'
      colormap(gray(256));
   end
end




% --- Executes on button press in Glasses.
function Glasses_Callback(hObject, eventdata, handles)
% hObject    handle to Glasses (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Glasses


% --- Executes during object creation, after setting all properties.
```

```
function figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes on button press in ClearFields.
function ClearFields_Callback(hObject, eventdata, handles)
% hObject    handle to ClearFields (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.Obstructed,'Value',0);
set(handles.Trauma,'Value',0);
set(handles.Disease,'Value',0);
set(handles.Glasses,'Value',0);
set(handles.Contacts,'Value',0);
set(handles.Surgery,'Value',0);
set(handles.SubjectNumber,'Value',0);
set(handles.SubjectNumber,'String','00000');
set(handles.Male,'Value',1);
set(handles.LeftEye,'Value',1);
set(handles.ImageNumber,'String','1');
set(handles.ImageNumber,'Value',1);
set(handles.angle0, 'Value', 1);




function IrisNumber_Callback(hObject, eventdata, handles)
% hObject    handle to IrisNumber (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of IrisNumber as text
%        str2double(get(hObject,'String')) returns contents of IrisNumber as a double


% --- Executes during object creation, after setting all properties.
function IrisNumber_CreateFcn(hObject, eventdata, handles)
% hObject    handle to IrisNumber (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes during object creation, after setting all properties.
function Male_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Male (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes during object creation, after setting all properties.
function angle0_CreateFcn(hObject, eventdata, handles)
% hObject    handle to angle0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes on button press in PreviewButton.
function PreviewButton_Callback(hObject, eventdata, handles)
% hObject    handle to PreviewButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global g_vidObjects
DeviceList=get(handles.DeviceList);
closepreview;
%vidRes = get(g_vidObjects, 'VideoResolution');
%nBands = get(g_vidObjects,'NumberOfBands');
%hImage = image(zeros(vidRes(2),vidRes(1),nBands));
preview(g_vidObjects(DeviceList.Value));
%colormap(gray(256));

% --- Executes during object creation, after setting all properties.
function IrisCaptureFigure_CreateFcn(hObject, eventdata, handles)
% hObject    handle to IrisCaptureFigure (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

function Subject=GetSubjectNumber( handles )
%
Subject=get(handles.SubjectNumber,'String');
```

```
function Eye=GetEye( handles )
% Return the Eye Selected
if ( get(handles.LeftEye,'Value') == 1 )
   Eye='L';
else
   Eye='R';
end

function Sex=GetSex( handles )
% Return the Sex of the Subject
if ( get(handles.Male,'Value') == 1 )
   Sex='m';
else
   Sex='f';
end

function Age=GetIrisAge( handles )
% Return the Age of the Iris
Age = get(handles.IrisAge,'String');

function Color=GetIrisColor( handles )
% Return the Color of the Iris
Color = get(handles.IrisColor,'String');
Color = cell2mat(Color(get(handles.IrisColor,'Value')));

function Angle = GetIrisAngle(handles)
if (get(handles.angle0,'Value') == 1)
   Angle = '0';
elseif (get(handles.angle15,'Value') == 1)
   Angle = '15';
elseif (get(handles.angle30, 'Value') == 1)
   Angle = '30';
elseif (get(handles.angle45,'Value') == 1)
   Angle = '45';
end


function ImageNumber=GetLastImageNumber( handles, Subject )
% Get the last image number of a given Subject or
% of the Subject Currently Selected
if (Subject=='')
   Subject = GetSubjectNumber(handles);
end
```

```
function DATESTR=GetDate( D )
% Return the current date
if ( D == '' )
   D = now;
end
DSTR=datestr(D,'yymmdd');

function Details=GetSubjectDetails(handles)
% return the results of the checkboxes
% Leaving room for 5 extra details we didn't think about yet.
% There has to be a better way of doing this!
if(get(handles.Obstructed,'Value'))
   Details=sprintf('O');
else
   Details=sprintf('');
end
if(get(handles.Glasses,'Value'))
   Details=sprintf('%s,G',Details);
else
   Details=sprintf('%s,',Details);
end
if(get(handles.Contacts,'Value'))
   Details=sprintf('%s,C',Details);
else
   Details=sprintf('%s,',Details);
end
if(get(handles.Trauma,'Value'))
   Details=sprintf('%s,T',Details);
else
   Details=sprintf('%s,',Details);
end
if(get(handles.Disease,'Value'))
   Details=sprintf('%s,D',Details);
else
   Details=sprintf('%s,',Details);
end
if(get(handles.Surgery,'Value'))
   Details=sprintf('%s,S,,,,',Details);
else
   Details=sprintf('%s,,,,,',Details);
end
```

```
% Make sure you delete the commas before you add more details!!!!


function SubjectNumber_Callback(hObject, eventdata, handles)
% hObject    handle to SubjectNumber (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of SubjectNumber as text
%        str2double(get(hObject,'String')) returns contents of SubjectNumber as a double


% --- Executes during object creation, after setting all properties.
function SubjectNumber_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SubjectNumber (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end




% --- Executes on button press in NextImage.
function NextImage_Callback(hObject, eventdata, handles)
% hObject    handle to NextImage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global IRIS_ROOT;
Subject=GetSubjectNumber(handles);
path=sprintf('%s\\%s',IRIS_ROOT,Subject);
curr=get(handles.ImageNumber,'Value');
curr=curr+1;
```

```
set(handles.ImageNumber,'Value',curr);
set(handles.ImageNumber,'String',curr);
if ( exist(path) == 7)

end



% --- Executes on button press in PreviousImage.
function PreviousImage_Callback(hObject, eventdata, handles)
% hObject    handle to PreviousImage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global IRIS_ROOT;
Subject=GetSubjectNumber(handles);
path=sprintf('%s\\%s',IRIS_ROOT,Subject);
curr=get(handles.ImageNumber,'Value');
curr=max(1,curr-1);
set(handles.ImageNumber,'Value',curr);
set(handles.ImageNumber,'String',curr);
if ( exist(path) == 7)

end

function UpdateImage( handles )
% update the Displayed image with one that exists if it exists.



% --- Executes on button press in VidCapture.
function VidCapture_Callback(hObject, eventdata, handles)
% hObject    handle to VidCapture (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in SaveImages.
function SaveImages_Callback(hObject, eventdata, handles)
% hObject    handle to SaveImages (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global IRIS_ROOT;
global IRIS_DB;
global g_video;
```

```
global FRAMES_PER_TRIGGER;
Subject=GetSubjectNumber(handles);
Eye=GetEye(handles);
Sex=GetSex(handles);
Age=GetIrisAge(handles);
Color=GetIrisColor(handles);
Angle=GetIrisAngle(handles);

Details=GetSubjectDetails(handles);

directory=GetDirectory( handles );
for a=1:FRAMES_PER_TRIGGER,

   filename=GetFilename( handles );
   if ( exist(filename) == 2 )
      button=questdlg('This file exists! \nAre you sure want to overwrite it?');
   else
      button = 'Yes';
   end
   if ( button == 'Yes')

      if ( exist(directory) == 7 )
         imwrite(g_video(:,:,1,a),filename,'bmp');
      else
         mkdir(directory);
         imwrite(g_video(:,:,1,a),filename,'bmp');
      end
      fid=fopen(sprintf('%s%s',IRIS_ROOT,IRIS_DB),'a');

imageinfo=sprintf('%s,%s,%s,%s,%s,%s,%s,%s',Subject,Eye,Sex,Age,Color,Details,Angle,filen
ame);
      fprintf(fid,'%s\n',imageinfo);
      fclose(fid);
      set(handles.FileName,'String',imageinfo);
   end
   NextImage_Callback(hObject, eventdata, handles)
end

% --- Executes on button press in CaptureVideo.
function CaptureVideo_Callback(hObject, eventdata, handles)
% hObject    handle to CaptureVideo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global FRAMES_PER_TRIGGER
```

```
global PERIOD;
global ACTUAL_FRAME_RATE
global DESIRED_FRAME_RATE
global g_vidObjects;
global g_drivers;
global g_video;
% FPS = 10,  save single files as bmps.  increment counter, both eyes
% with/without glasses contacts if available. distinguish filename by
% N - Normal eye,  G - Glasses  C - Contacts
DeviceList=get(handles.DeviceList);
vidObj=g_vidObjects(DeviceList.Value);
oldFramesPerTrigger=get(vidObj,'FramesPerTrigger');
set(vidObj,'FramesPerTrigger',FRAMES_PER_TRIGGER);
oldGrabInterval=get(vidObj,'FrameGrabInterval');
set(vidObj,'FrameGrabInterval',ACTUAL_FRAME_RATE/DESIRED_FRAME_RATE);
start(vidObj);
%index=floor(linspace(1,FRAMES_PER_TRIGGER*ACTUAL_FRAME_RATE/DESIRED_F
RAME_RATE,FRAMES_PER_TRIGGER))
[g_video,t]=getdata(vidObj);
%g_video=g_video_tmp(:,:,:,index);  % subsample (sort of)
imaqmontage(g_video);
stop(vidObj);
% Restore old settings.
set(vidObj,'FrameGrabInterval',oldGrabInterval);
set(vidObj,'FramesPerTrigger',oldFramesPerTrigger);
```

Appendix C: Experimental Data

Experiment 1: Orthogonal Enrollment Templates

	Percent Accuracy				
Rank	**0°**	**15°**	**30°**	**45°**	**All**
1	61.2368	18.3284	10.8025	5.2482	23.6471
2	67.1192	25.0733	17.1296	9.5035	29.4292
3	69.3816	30.6452	20.8333	12.1986	32.9874
4	70.8899	34.8974	24.2284	13.9007	35.6931
5	72.6998	37.2434	27.7778	16.4539	38.2506
6	73.6048	38.4164	30.0926	18.156	39.7702
7	74.6606	39.8827	31.9444	20	41.3269
8	75.264	41.6422	34.8765	22.5532	43.2913
9	76.3198	43.4018	38.2716	24.539	45.3299
10	76.4706	45.0147	39.6605	26.6667	46.6642
11	77.2247	46.3343	41.2037	29.2199	48.2209
12	77.9789	48.2405	43.2099	30.7801	49.7776
13	79.1855	49.7067	43.9815	32.766	51.149
14	79.7888	51.3196	45.216	35.0355	52.5945
15	80.2413	53.0792	46.4506	36.8794	53.9288
16	80.2413	54.3988	48.1481	38.4397	55.0778
17	81.5988	56.0117	49.537	39.8582	56.5234
18	82.0513	57.1848	50.3086	40.8511	57.3758
19	83.1071	58.5044	50.9259	41.7021	58.3395
20	83.2579	59.5308	51.5432	42.9787	59.1179
21	83.5596	60.9971	52.4691	44.1135	60.0815
22	83.8612	61.7302	54.1667	44.9645	60.9711
23	84.0121	62.61	55.4012	45.8156	61.7494
24	84.4646	63.3431	56.4815	46.8085	62.5649
25	84.7662	63.4897	57.0988	47.3759	62.9726
26	85.0679	63.9296	57.2531	48.227	63.4173
27	85.0679	65.5425	58.179	49.2199	64.3069
28	85.3695	66.2757	59.2593	49.6454	64.937
29	85.6712	66.5689	60.3395	50.3546	65.53
30	86.1237	67.1554	60.9568	51.2057	66.1601
31	86.2745	68.0352	61.4198	51.9149	66.7161
32	86.5762	68.3284	61.8827	52.766	67.1979
33	86.5762	69.0616	62.963	53.7589	67.9021
34	86.8778	69.5015	64.0432	54.7518	68.6064
35	87.1795	69.7947	64.6605	55.1773	69.0141
36	87.3303	70.2346	65.1235	55.8865	69.4589
37	87.4811	70.3812	65.7407	56.5957	69.8666
38	87.7828	70.5279	65.8951	57.4468	70.2372
39	88.0845	71.261	66.358	58.0142	70.7561
40	88.2353	71.8475	66.5123	58.7234	71.1638
41	88.3861	72.1408	66.6667	59.7163	71.5715
42	88.537	72.2874	67.4383	59.8582	71.8681
43	88.8386	72.7273	68.5185	60.5674	72.4981
44	89.1403	72.8739	70.0617	60.5674	72.98
45	89.1403	73.1672	70.8333	60.8511	73.3136
46	89.2911	73.3138	71.7593	61.7021	73.8325
47	89.2911	73.4604	72.2222	62.2695	74.129
48	89.5928	73.7537	72.5309	63.1206	74.5738
49	89.5928	74.0469	73.1481	63.5461	74.9073
50	89.5928	74.9267	73.6111	63.6879	75.278

Experiment 2: 15° Enrollment Templates

	Percent Accuracy				
Rank	**0°**	**15°**	**30°**	**45°**	**All**
1	18.2504	65.5425	14.3519	4.8227	25.7598
2	27.3002	71.8475	23.3025	11.3475	33.4322
3	35.4449	74.3402	27.4691	14.1844	37.8058
4	39.3665	75.6598	33.9506	16.3121	41.2157
5	43.8914	76.2463	37.3457	18.4397	43.8473
6	47.0588	76.9795	41.2037	21.1348	46.4418
7	50.6787	78.8856	44.2901	23.8298	49.2587
8	53.2428	79.7654	47.5309	25.5319	51.3343
9	55.2036	80.3519	50.6173	27.234	53.1505
10	56.5611	80.6452	51.6975	29.6454	54.4477
11	57.9186	80.7918	53.5494	32.0567	55.8933
12	59.1252	81.6716	55.5556	33.0496	57.1534
13	60.181	81.8182	57.5617	35.7447	58.636
14	60.9351	81.9648	60.0309	37.7305	59.9703
15	62.5943	82.2581	60.1852	39.5745	60.9711
16	64.2534	82.2581	60.9568	41.1348	61.9718
17	65.1584	82.4047	62.3457	41.844	62.7502
18	65.7617	82.8446	63.1173	42.8369	63.4544
19	66.5158	83.1378	63.8889	44.2553	64.2698
20	67.4208	83.4311	64.8148	45.2482	65.0482
21	68.175	83.4311	66.0494	46.6667	65.9007
22	69.6833	83.5777	66.9753	48.6525	67.0497
23	70.2866	83.7243	67.5926	49.3617	67.5686
24	70.4374	84.0176	67.9012	50.6383	68.0875
25	71.0407	84.1642	69.1358	51.3475	68.7546
26	72.3982	84.4575	69.7531	52.0567	69.4959
27	73.0015	84.7507	69.9074	53.0496	70.0148
28	73.7557	84.8974	70.216	53.9007	70.5337
29	74.0573	85.044	70.679	54.3262	70.8673
30	74.359	85.3372	70.8333	55.461	71.3491
31	74.8115	85.4839	71.6049	55.6028	71.7198
32	74.8115	85.6305	72.0679	56.1702	72.0163
33	75.1131	85.7771	72.2222	56.3121	72.2016
34	75.5656	86.0704	72.5309	57.0213	72.6464
35	76.0181	86.217	72.6852	57.4468	72.9429
36	76.0181	86.3636	73.3025	57.7305	73.2024
37	76.1689	86.3636	73.9198	57.7305	73.3877
38	76.1689	86.6569	74.0741	58.2979	73.6471
39	76.3198	86.9501	74.3827	58.7234	73.9437
40	76.3198	87.0968	74.6914	59.8582	74.3514
41	76.3198	87.2434	74.8457	60.2837	74.5367
42	77.0739	87.2434	75.3086	60.5674	74.9073
43	77.2247	87.39	76.0802	60.9929	75.278
44	77.5264	87.8299	76.3889	61.7021	75.7228
45	77.5264	88.1232	76.5432	62.2695	75.9822
46	78.2805	88.2698	76.6975	62.4113	76.2787
47	78.733	88.2698	77.1605	62.5532	76.5382
48	78.733	88.2698	77.6235	62.9787	76.7606
49	78.733	88.2698	77.7778	63.2624	76.8718
50	78.733	88.4164	77.9321	63.8298	77.0941

Experiment 3: 30° Enrollment Templates

	Percent Accuracy				
Rank	**0°**	**15°**	**30°**	**45°**	**All**
1	12.0664	17.1554	63.2716	13.4752	26.0193
2	18.0995	24.7801	67.9012	18.8652	31.9496
3	24.2836	29.7654	69.9074	22.1277	36.0638
4	28.9593	34.0176	72.2222	25.1064	39.6219
5	31.9759	36.9501	73.7654	27.0922	41.9941
6	34.8416	39.5894	75.1543	28.9362	44.1809
7	37.7074	41.0557	76.3889	31.6312	46.2565
8	40.1207	43.5484	77.6235	35.0355	48.6657
9	41.9306	46.0411	78.5494	36.5957	50.3706
10	43.5897	49.2669	79.6296	38.4397	52.3351
11	44.9472	51.6129	80.5556	41.2766	54.2254
12	47.0588	53.2258	81.4815	43.4043	55.9303
13	48.4163	54.9853	82.0988	44.9645	57.2646
14	50.0754	56.5982	82.716	46.2411	58.5619
15	52.3379	57.6246	83.179	48.227	60.0074
16	52.7903	59.9707	83.3333	50.3546	61.3047
17	53.6953	61.1437	84.1049	52.0567	62.4537
18	54.7511	61.8768	84.4136	53.3333	63.3062
19	55.6561	63.6364	84.7222	55.0355	64.4922
20	56.2594	64.5161	85.1852	55.8865	65.1964
21	57.1644	65.5425	85.6481	56.5957	65.9748
22	58.8235	66.5689	86.8827	57.4468	67.1609
23	59.4268	68.0352	87.3457	58.7234	68.1245
24	60.3318	68.7683	87.963	60.2837	69.0882
25	61.086	69.3548	88.1173	61.1348	69.6812
26	61.6893	69.9413	88.7346	62.2695	70.4225
27	62.8959	70.5279	89.1975	63.5461	71.3121
28	63.4992	70.8211	89.5062	64.3972	71.831
29	63.9517	71.4076	90.1235	64.9645	72.387
30	65.9125	71.8475	90.1235	65.5319	73.1282
31	66.6667	73.1672	90.8951	66.6667	74.129
32	67.5716	74.1935	90.8951	68.0851	74.9815
33	68.4766	74.9267	90.8951	68.5106	75.5004
34	69.6833	75.5132	90.8951	69.6454	76.2417
35	70.7391	76.2463	91.0494	70.0709	76.8347
36	71.9457	76.6862	91.2037	70.4965	77.3907
37	73.1523	76.9795	91.5123	71.3475	78.0578
38	73.9065	77.566	91.9753	72.0567	78.6879
39	74.9623	77.8592	92.284	72.3404	79.1698
40	76.0181	78.4457	92.4383	72.9078	79.7628
41	76.7722	78.8856	92.5926	73.617	80.2817
42	78.2805	79.3255	92.7469	73.7589	80.8377
43	79.3363	79.6188	93.0556	74.0426	81.3195
44	79.638	79.912	93.2099	74.4681	81.616
45	80.0905	79.912	93.2099	75.0355	81.8755
46	80.3922	80.2053	93.2099	75.6028	82.172
47	80.8446	80.3519	93.2099	76.1702	82.4685
48	80.9955	80.4985	93.3642	76.3121	82.6168
49	81.9005	80.7918	93.5185	77.0213	83.1357
50	82.5038	80.7918	93.5185	77.4468	83.3951

Experiment 4: 45° Enrollment Templates

	Percent Accuracy				
Rank	**0°**	**15°**	**30°**	**45°**	**All**
1	6.0332	9.3842	10.9568	49.7872	19.4959
2	12.6697	14.8094	16.9753	56.8794	25.7969
3	17.3454	18.0352	23.4568	60.8511	30.3558
4	21.1161	20.6745	26.3889	64.8227	33.6916
5	24.4344	24.4868	30.2469	67.5177	37.1016
6	26.546	28.0059	34.2593	69.5035	39.9926
7	29.1101	31.3783	37.3457	69.9291	42.3277
8	31.0709	33.2845	40.2778	72.0567	44.5515
9	33.9367	36.9501	42.9012	73.3333	47.146
10	35.5958	40.0293	44.4444	74.1844	48.9251
11	37.1041	42.6686	46.142	75.3191	50.6672
12	39.5173	45.0147	47.9938	76.1702	52.5204
13	41.0256	47.3607	50	77.0213	54.1883
14	43.1373	48.9736	50.7716	77.7305	55.4855
15	43.7406	50.2933	53.5494	79.4326	57.0793
16	45.3997	51.6129	55.8642	79.8582	58.4878
17	47.0588	53.5191	57.716	80.2837	59.9333
18	48.2655	55.132	58.7963	81.1348	61.1193
19	49.1704	55.8651	59.5679	81.9858	61.9348
20	49.9246	57.0381	60.1852	82.4113	62.6761
21	51.1312	58.0645	61.5741	83.4043	63.8251
22	52.3379	58.5044	62.963	83.5461	64.6034
23	54.6003	60.4106	64.6605	83.5461	66.0489
24	56.4103	61.2903	65.8951	84.3972	67.235
25	57.7677	61.8768	67.284	85.1064	68.2357
26	59.276	62.9032	67.7469	85.8156	69.1623
27	60.6335	63.6364	68.3642	86.0993	69.9036
28	61.086	64.0762	69.2901	86.6667	70.4967
29	61.991	64.956	70.5247	87.0922	71.3491
30	62.4434	65.8358	70.9877	87.0922	71.7939
31	63.8009	66.2757	71.4506	87.5177	72.4611
32	65.1584	66.4223	72.3765	87.8014	73.1282
33	65.9125	66.8622	72.8395	88.0851	73.6101
34	66.365	67.3021	73.4568	88.3688	74.0549
35	66.8175	67.7419	73.6111	88.9362	74.4626
36	67.5716	68.0352	74.537	89.078	74.9815
37	68.9291	68.6217	74.8457	89.9291	75.7598
38	70.5882	68.7683	75.9259	90.0709	76.5011
39	70.7391	69.5015	76.8519	90.6383	77.0941
40	71.644	70.2346	77.0062	90.7801	77.576
41	72.549	70.3812	77.6235	91.0638	78.0578
42	72.6998	70.5279	78.0864	91.3475	78.3173
43	73.1523	70.9677	78.2407	91.4894	78.6138
44	73.9065	71.4076	78.7037	91.773	79.0956
45	74.2081	71.8475	78.858	92.0567	79.3921
46	74.6606	72.1408	79.0123	92.1986	79.6516
47	74.9623	72.2874	79.4753	92.4823	79.9481
48	74.9623	72.7273	80.0926	92.6241	80.2446
49	75.4148	73.3138	80.2469	92.6241	80.5411
50	75.8673	74.4868	80.7099	92.6241	81.06

Experiment 5: Mixed Enrollment Templates

	Percent Accuracy				
Rank	**0°**	**15°**	**30°**	**45°**	**All**
1	29.5626	39.7361	37.1914	21.5603	31.8755
2	41.3273	51.0264	47.2222	30.6383	42.4018
3	46.908	56.1584	52.4691	35.461	47.5908
4	52.3379	62.3167	56.6358	40.5674	52.8169
5	55.2036	65.2493	59.4136	46.383	56.4492
6	58.6727	69.7947	61.1111	48.6525	59.4514
7	60.9351	73.0205	64.3519	51.9149	62.4537
8	64.4042	74.6334	67.284	54.7518	65.1594
9	66.5158	75.5132	69.2901	56.1702	66.7532
10	68.3258	76.9795	71.142	58.2979	68.5693
11	70.2866	78.1525	72.9938	60.2837	70.3113
12	72.0965	79.912	74.2284	61.9858	71.9422
13	74.5098	81.5249	75.463	63.9716	73.7583
14	75.5656	82.5513	76.3889	65.8156	74.9815
15	75.8673	84.1642	77.6235	66.8085	76.0193
16	76.9231	85.3372	78.5494	68.6525	77.2795
17	78.4314	86.8035	79.9383	69.9291	78.6879
18	78.8839	87.5367	81.0185	70.7801	79.4663
19	79.9397	88.563	82.0988	71.9149	80.5411
20	80.8446	89.2962	82.716	72.766	81.3195
21	81.9005	90.6158	83.4877	74.1844	82.4685
22	82.8054	91.7889	84.5679	76.0284	83.7287
23	84.0121	92.3754	85.6481	76.7376	84.6182
24	84.1629	92.8152	86.4198	77.5887	85.1742
25	84.7662	93.2551	86.8827	78.0142	85.656
26	85.0679	93.2551	87.1914	78.7234	85.9896
27	85.822	93.4018	87.6543	79.5745	86.5456
28	86.1237	93.4018	88.2716	80.7092	87.0645
29	86.727	93.8416	88.7346	81.844	87.7317
30	87.1795	93.8416	89.6605	82.4113	88.2135
31	87.3303	93.9883	89.6605	82.9787	88.4359
32	87.7828	94.1349	90.1235	83.1206	88.7324
33	88.537	94.2815	90.4321	83.5461	89.1401
34	88.8386	94.2815	91.0494	84.3972	89.5849
35	89.1403	94.5748	91.5123	85.1064	90.0297
36	89.5928	94.5748	91.9753	85.5319	90.3632
37	89.8944	94.868	92.284	86.5248	90.8451
38	90.6486	95.4545	92.4383	87.0922	91.364
39	90.9502	95.7478	93.2099	87.3759	91.7717
40	90.9502	96.0411	93.6728	87.3759	91.957
41	91.2519	96.1877	93.6728	87.5177	92.1053
42	91.7044	96.6276	93.8272	87.9433	92.4759
43	92.006	97.0674	94.1358	88.5106	92.8836
44	92.3077	97.5073	94.2901	88.7943	93.1801
45	92.7602	97.5073	94.4444	89.7872	93.5878
46	92.7602	97.5073	94.5988	89.9291	93.662
47	93.2127	97.5073	94.9074	90.6383	94.0326
48	93.5143	97.5073	95.0617	91.4894	94.3662
49	93.6652	97.654	95.679	91.773	94.6627
50	94.1176	97.654	95.8333	92.0567	94.8851

Appendix D: Publications

1. Ruth M. Gaunt, "Collection of Non-Orthogonal Iris Images for Iris Recognition," *2006 National Conference on Undergraduate Research* (6-8 April 2006).

2. Robert.W Ives, Lauren Kennell, Ruth M. Gaunt, D.M. Etter, "Iris Segmentation for Recognition Using Local Statistics," ", *IEEE 39th Annual Asilomar Conference on Signals, Systems, and Computers,* Nov. 2005.

Proceedings of the National Conference

On Undergraduate Research (NCUR) 2006
The University of North Carolina at Asheville
Asheville, North Carolina
April 6-8, 2006

Collection of Non-Orthogonal Iris Images for Iris Recognition

Ruth Gaunt
Electrical Engineering Department
United States Naval Academy
105 Maryland Avenue
Annapolis, MD 21402-5025. USA

Faculty Advisors: R.W. Ives, D.M. Etter

Abstract

Despite its high recognition rate, one of iris recognition's major weaknesses is that it requires that the users be fully cooperative when it comes to making sure their eye is close enough to the camera and is still enough for a high quality iris image to be collected. Current commercial systems require the iris to be orthogonal (looking directly into the camera) since their recognition algorithm must first detect the pupil, which is assumed to be a circle. This is only the case if the eye is looking directly at the camera lens. This makes it difficult or impossible for identification to occur if the image is taken from a non-orthogonal angle. A non-orthogonal iris image is defined as an image where the iris is not looking directly into the camera. This research involves devising a method to collect and organize a database of non-orthogonal iris images. The non-orthogonal iris image collection station allows iris images to be obtained at 0° (orthogonal), 15°, 30°, and 45° for each eye. The results of this research will aid in the development of an algorithm that can use non-orthogonal images for iris recognition.

1. Introduction

Biometrics is the study of the individual physical traits of a person that can be quantified and used for identification. Examples of different types of biometrics include fingerprints, hand geometry, face, voice, and iris. These quantifiable features are measured and stored in a database to be used for automatic recognition. The increased use of biometrics as a method for human identification has led to a decreased need for personal identification numbers (PIN) and passwords, which are easily spoofed. Using biometrics (such as the iris) leads to increased confidence that "imposters" do not gain access to resources, systems, or information that they are not authorized to access.

The iris is the colored portion of the eye that surrounds the pupil and controls the amount of light that enters the eye. It is the only internal human organ that can be observed in the external environment and is made up of tissue that lies behind the cornea [1]. Iris tissue patterns are formed as a part of fetal development, which involves random tearing of the iris tissue. Because iris patterns are not affected by genetics, no two people share the same iris patterns. In fact, the right and left eyes of a single person have different patterns [1]. By the time a person reaches one year of age, iris patterns have stabilized and will stay the same for a lifetime, excluding any major eye injuries or disease that may occur [1]. Iris recognition algorithms quantify these highly variable patterns and use them for identification.

Verification and identification are the two most common applications of biometric systems, including iris recognition. Verification is a one-to-one match, meaning, for example, that an individual enters a PIN while presenting a biometric, such as a fingerprint or iris, at the same time [2]. A positive match occurs if the person whose biometric data is presented matches the person whose PIN was entered. Identification, on the other hand, is a one-to-many match [2]. For example, this means that an individual who wants to gain access to a secure location

looks directly into an iris camera, and his or her iris patterns are compared to all of the iris patterns in a given database to check for a positive match. If the individual's iris patterns meet a certain threshold, identification occurs and access is granted.

One other application of biometric technology, which is a much more difficult problem, involves a many-to-many search, also referred to as a "watchlist" [2]. In this scenario, a large area such as an airport is scanned in order to check for individuals of interest (i.e. terrorists and felons). The biometric information of these individuals is stored in a database known as a watchlist, and all individuals who pass a given checkpoint have their biometric data collected and compared to the watchlist, typically under covert conditions. The typically large size of these databases as well as the covert collection conditions make this a complex problem to solve.

This project helps to address the problem of covert iris recognition. Even though iris recognition has a very high identification rate, one of its major drawbacks is that it requires the user to be fully cooperative when it comes to making sure that the eye is close enough to the camera and still enough to have a clear image captured. Near-infrared (NIR) cameras are used in iris recognition because iris patterns stand out more under NIR illumination (790 nm). Current commercial recognition systems require the user to stare straight into the camera so that an orthogonal image is captured. Orthogonal iris capture is necessary because recognition algorithms typically must first detect the inner (pupillary) and outer (limbic) boundaries of the iris, which are most often assumed to be circles, and this is only true when the subject is staring directly into the camera lens. This means that in covert situations where subjects are not staring directly into a camera (non-orthogonal), identification cannot occur because the pupillary and limbic boundaries of the iris are now ellipses instead of circles and cannot be detected (Figure 1). The main purpose of this research is to develop a database of non-orthogonal iris images taken from four known angles to aid in the development and testing of non-orthogonal iris recognition algorithms.

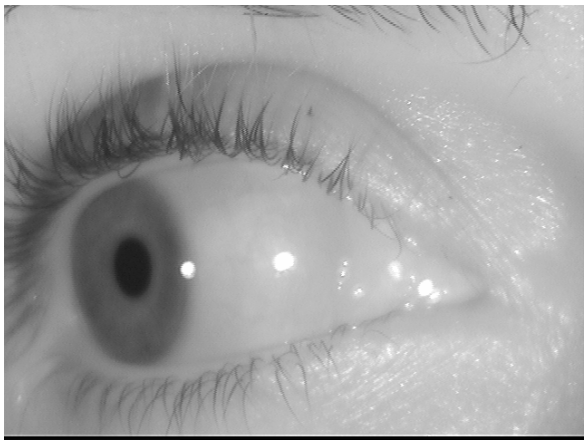

Figure 1. Non-orthogonal iris image.

2. Methodology

In order to provide for the accurate collection of non-orthogonal iris images at known orientation angles, a collection station was built that allows the user's head to remain stationary throughout the collection process. The iris camera moves around the user's head on a track (Figure 2). The database of non-orthogonal iris images contains images taken at four known orientation angles: 0° (orthogonal), 15°, 30°, and 45°. First, the user places his or her chin in the chin rest so that the head remains in one place. The chin rest can be raised or lowered so that no matter what the proportions of a person's face are, the eye can always be positioned to be in the center of the camera lens. There are two thin metal rods at the opposite end of the collection station for the user to focus on so that the only angle variation that occurs during the collection process comes from changes in camera position and not the shifting of the users' eyes in the sockets. The camera is on a raised platform that moves on a track. It is held in place by a pin that fits into holes placed at the desired collection angles for each eye. In addition, the collection station was constructed so that the distance from the camera to the eye is five inches, which is the desirable distance for achieving an

optimal level of focus so that enough iris pattern information is available in each image. The high-quality near-infrared camera that is used is the LG IrisAccess® 3000 entry control system.

Figure 2. Non-orthogonal iris image collection station.

An existing USNA iris collection graphical user interface (GUI) was altered so that information such as the angle at which the image is obtained is stored along with other information about the individual when the iris image is saved (Figure 3). This information includes user subject number, which eye is being collected (right or left), gender, iris color, iris age, and whether the individual is wearing glasses, contacts, and if the user has a history of eye trauma or eye surgery [3]. For purposes of this research, users are instructed to remove their glasses so that changes in iris patterns due to optical distortion by glass lenses is not a variable.

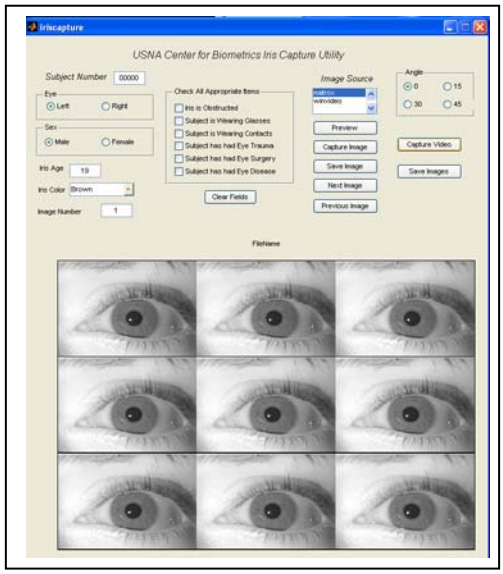

Figure 3. Graphical user interface for iris collection.

The iris camera is used in conjunction with the MATLAB Image Acquisition and Image Processing Toolboxes and the Matrox Meteor II frame grabber to collect the data [3]. Nine frames of video are grabbed at a time, and all nine images are saved. This means that for each eye, thirty-six images are obtained, since there are four different orientation angles and nine images are saved for each of these four angles. Figure 4 shows examples of images from each of the four orientation angles.

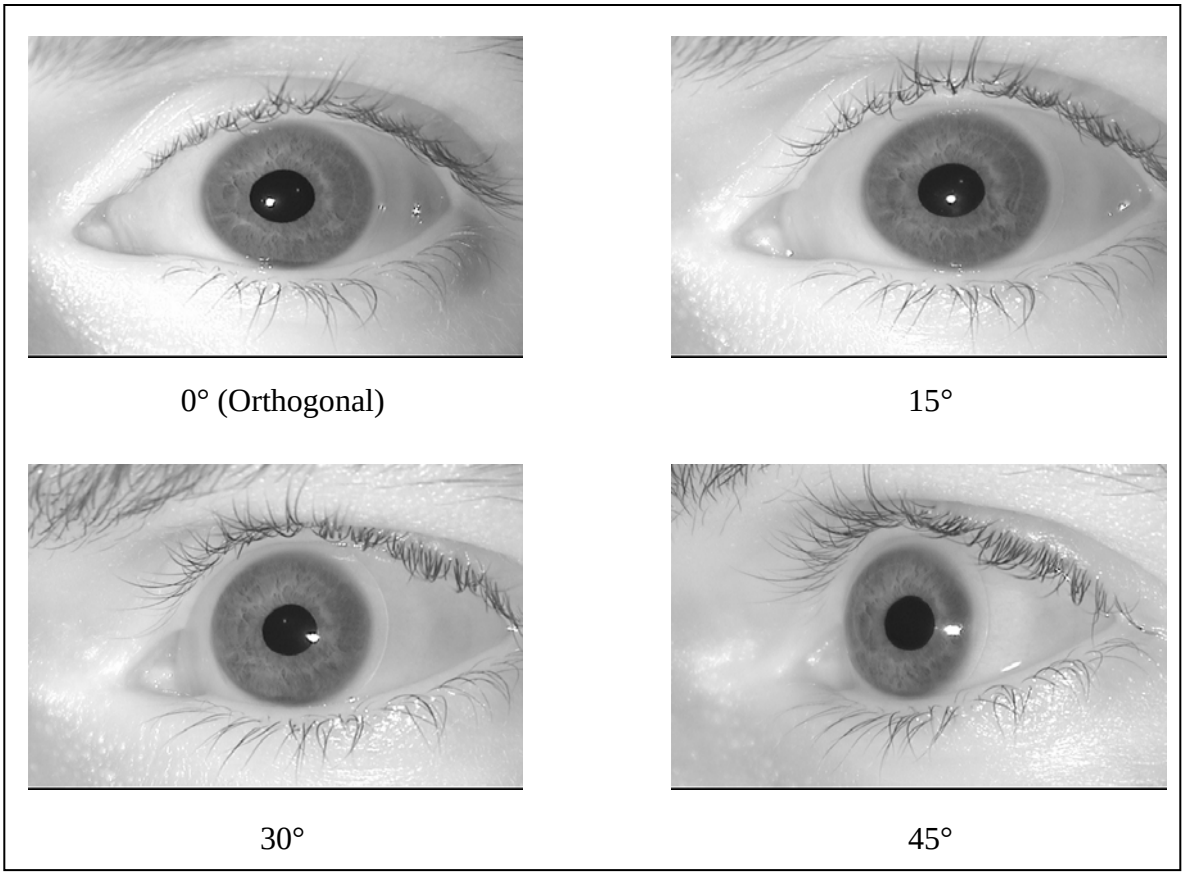

Figure 4. Database images from each non-orthogonal angle.

3. Data

Data for approximately ninety irises are stored in the non-orthogonal database. These irises were collected at each non-orthogonal angle, and about sixty of those went through two collections over the course of a semester resulting in a database of almost 5000 images.

After collection, the database images were run through an existing non-orthogonal iris segmentation algorithm that outputs the parameters of elliptical boundaries of the pupillary and limbic boundaries, including the centroid, semi-major axis, and semi-minor axis [4]. To assess the accuracy of collection at each non-orthogonal angle, the ratio of the semi-major axis to semi-minor axis of the pupillary boundary was calculated for each subject eye. Since the eccentricity of the pupillary boundary increases as the non-orthogonal imaging angle increases, the ratio of semi-major axis to semi-minor axis of the pupillary boundary should increase as well. Table 1 displays the mean ratio and standard deviation for each non-orthogonal angle. Images taken from an angle of zero degrees (orthogonal images) have the smallest mean ratio of 1.085. This is to be expected because the entire iris can be seen in the image, and in general, the limbic and pupillary boundaries of irises are approximately circular, which would translate to equal

semi-major and semi-minor axes (ratio = 1.0). As the non-orthogonal imaging angle increases, the visible iris boundaries become more and more elliptical, and at an angle of 45°, the mean ratio was 1.3668.

Figure 5 shows a histogram of the semi-major axis/semi-minor axis ratio values for images collected at the four different angles. This graph shows that there is much overlap between the different non-orthogonal angles. In fact, the orthogonal and 15° degree images are virtually indistinguishable. Despite the overlap, the peak ratios for the 30° and 45° iris images are at increasingly higher ratios, which is expected.

Database Collection Analysis		
Angle	Mean Ratio	Standard Deviation
0°	1.085	0.3861
15°	1.1157	0.375
30°	1.2147	0.3492
45°	1.3668	0.4227

Table 1. Iris data used for database analysis.

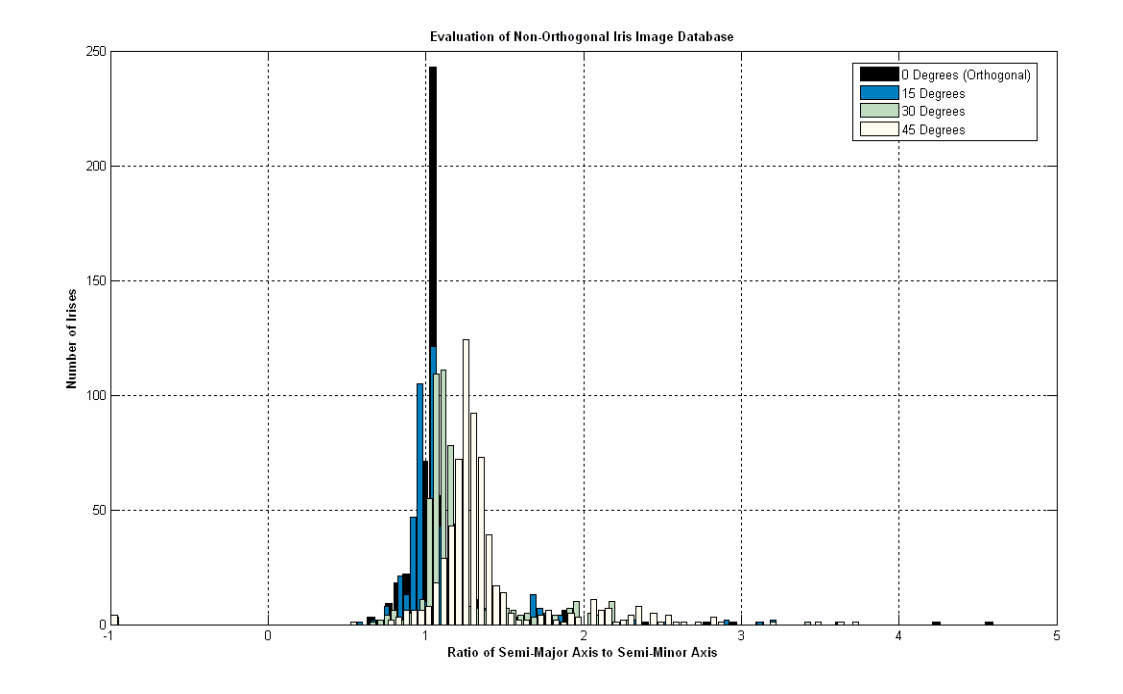

Figure 5. Analysis of non-orthogonal iris image database.

4. Conclusion

One of the difficulties with biometrics research is finding enough data, such as iris images, for testing. In the case of non-orthogonal iris recognition, there are presently only a few databases of non-orthogonal iris images, which make it difficult to develop robust recognition algorithms. This research has successfully produced a database of almost 5000 images.

The variations in collection results displayed by the ratios of semi-major axis to semi-minor axis may have occurred for three reasons. First, even though the subject is instructed to stare straight ahead during the collection

process and has visual aids at which to stare, there is no guarantee that the person was looking straight ahead at the instant the images were captured. In addition, the performance of the segmentation algorithm that finds the parameters of the elliptical iris boundaries is not perfect, and the true location of the boundaries is subjective anyway. In addition, non-orthogonal iris images are not perfect ellipses because human iris shapes are not always perfectly circular, even in orthogonal images.

5. Acknowledgements

The author wishes to express her appreciation to:
Dr. Robert Ives, Electrical Engineering Department, USNA – Primary Project Adviser
Dr. Delores Etter, Electrical Engineering Department, USNA –Secondary Project Adviser
Dr. Lauren Kennell, Electrical Engineering Department, USNA-Research Asst. Professor
LT Robert Schultz, USN, Electrical Engineering Department, USNA
Mr. Jerry Ballman, Electrical Engineering Department, USNA- Laboratory Technician
Mr. Michael Wilson, Electrical Engineering Department, USNA – Laboratory Technician

6. References

[1] Y. Du, R. W. Ives, and D. M. Etter, "Iris Recognition", *The Electrical Engineering Handbook*, 3rd Edition, Boca Raton, FL: CRC Press, 2004 (in press).
[2] Y. Du, R. W. Ives, D. M. Etter, T. B. Welch, and C.-I Chang, "One Dimensional Approach to Iris Recognition", *Proceedings of the SPIE*, pp. 237-247, Apr., 2004.
[3] R. Schultz, R.W. Ives and D.M. Etter, "Biometric Data Acquisition using MATLAB GUIs," *IEEE Frontiers in Education 2005*, Oct. 2005.
[4] B. Bonney, "Non-Orthogonal Iris Localization," *Final Trident Report* Apr. 2005.

7. Works Consulted

1. Calvert, J.B. "Ellipse," *Dr. James B. Calvert.* 31 May 2005. <http://www.du.edu/~jcalvert/math/ellipse.htm>.

2. Daugman, John. *How Iris Recognition Works.* 25 Oct 2003. <www.cl.cam.ac.uk/users/jgd1000/irisrecog.pdf>.

3. Y. Du, B. Bonney, R.W. Ives, D.M. Etter and R. Schultz, "Partial Iris Recognition using a 1-D Approach: Statistics and Analysis," *2005 IEEE International Conference on Acoustics, Speech and Signal Processing*, Philadelphia, Mar 2005.

4. Y. Du, R.W. Ives, R. Schultz and D.M. Etter, "Analysis of Partial Iris Recognition," *2005 SPIE Defense and Security Symposium*, Orlando, FL, Mar-Apr 2005.

5. Weisstein, Eric W. "Ellipse." From *MathWorld*--A Wolfram Web Resource. 24 May 2005. <http://mathworld.wolfram.com/Ellipse.html>

Iris Segmentation for Recognition Using Local Statistics

Robert W. Ives, Lauren R. Kennell, Ruth M. Gaunt and Delores M. Etter
Electrical Engineering Department, U.S. Naval Academy
Annapolis, MD 21402-5025

***Abstract*- Iris recognition is one of the more commonly used biometrics for recognition due to its accuracy. One of the first steps in iris recognition is to segment the iris from the rest of the image for further processing. This paper investigates the use of local statistics to find the pupillary (inner) and limbic (outer) boundaries of the iris. In particular, the local standard deviation and local kurtosis have shown to be useful in this respect. The methodology and results using the University of Bath and the United States Naval Academy iris databases are presented.**

I. INTRODUCTION

The iris is the colored portion of the eye that surrounds the pupil. It consists of richly textured patterns that are distinct from person to person, and in fact are distinct from left eye to right eye from the same person. Iris recognition systems are non-invasive to their users, but commercial systems do require a cooperative subject. For this reason, iris recognition is usually used for verification or identification purposes, rather than, for example, covert surveillance.

Figure 1 is an example of an iris image, which typically includes the eye as well as eyelids, eyelashes and perhaps part of the forehead and nose. Since the only information used in iris recognition are pixels that actually fall on the iris, one of the first steps in preprocessing the image before extracting its unique features is to segment the iris from the rest of the image. Several means have been proposed to perform this segmentation [1]-[5], but this paper investigates a different approach using local statistics to aid in determining the boundaries of the iris.

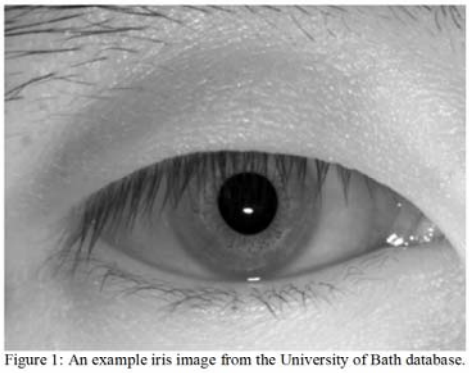

Figure 1: An example iris image from the University of Bath database.

Images from two iris databases were used in this research. First, we use the near infrared (NIR) iris image database collected by the University of Bath, U.K [6]. This database consists of 1000 images from 25 subjects (50 irises, 20 images per iris). These images are high resolution (1280 x 960 pixels), and have been compressed with JPEG-2000 to 0.5 bits/pixel. Second, we use the U.S. Naval Academy's in-house NIR iris database. This database consists of approximately 3000 images from 102 subjects, and is of standard video resolution, 640 x 480 pixels.

II. LOCAL STATISTICS

There are several parameters of interest in this research, including those which incorporate local statistics using higher-order central moments. *Local* here refers to the parameters that characterize small neighborhoods about each pixel throughout an image. These parameters include: the variance (or rather the standard deviation), skewness and kurtosis. These parameters typically are used to describe the probability distribution of sample data, and here are used to characterize the distribution of values in neighborhoods about each pixel in an image of an iris.

We begin by defining some terms related to statistical data analysis. First, the k^{th} moment of a set of data samples is defined as:

$$M_k = E(X^k), \tag{1}$$

where E is the expectation operator and the data values are identified by the variable X. The 1st moment is the mean value, $\overline{X}$. The k^{th} *central* moment is a moment taken about the mean and is defined as:

$$\mu_k = E(X - \overline{X})^k. \tag{2}$$

In particular, the 2nd central moment is called the variance, σ^2:

$$\sigma^2 = E\left(X - \overline{X}\right)^2. \tag{3}$$

The standard deviation, σ, is the square root of the variance and is used to determine the skewness and the kurtosis.

The *skewness* uses both the 3rd central moment and the standard deviation and is defined as:

$$\tau = \frac{\mu_3}{\sigma^3} = \frac{E\left(X - \bar{X}\right)^3}{\sigma^3}. \quad (4)$$

Skewness is a measure of the asymmetry of the data values relative to the mean value. If skewness is negative, more of the data have values smaller than the mean (that is, the distribution of values is more spread out to the left of the mean), and if skewness is positive, more of the values are larger than the mean.

Finally, *kurtosis* is the 4th central moment divided by the standard deviation to the fourth power, and is defined as:

$$\kappa = \frac{\mu_4}{\sigma^4} = \frac{E\left(X - \bar{X}\right)^4}{\sigma^4}. \quad (5)$$

Kurtosis is a measure of how peaked the data is. The kurtosis of a standard normal distribution of values (with standard deviation 1.0) is 3.0, and distributions that are "flatter" and less prone to outliers have kurtosis greater than 3 while those distributions that are more peaked and more prone to outliers than a standard normal distribution have kurtosis greater than 3.0.

III. SEGMENTATION

It is assumed that in local neighborhoods of fairly homogenous texture such as the pupil or the sclera (the white part of the eye) the relative variation of values is small so that the central moments will be relatively small. This means that the standard deviation, skewness and kurtosis will also be small. However, when neighborhoods about pixels cover distinct boundaries, such as one which covers both the dark of the pupil and the lighter shades of gray of the iris, the information that is provided by the standard deviation, skewness and kurtosis may in fact help define boundaries of the iris.

These parameters are computed as follows. For each pixel x in the original image, a small neighborhood of $n \times n$ pixels is extracted, centered at pixel x. Using these pixel values, the specific statistical parameters are computed for that neighborhood and the original pixel value is replaced by the new parameter. For example, using a 3×3 standard deviation window, 9 pixels are used to compute the local standard deviations. A new image is created that reflects the local standard deviations about each pixel in the original image.

Fig. 2 is the image in Fig. 1 after applying a 3x3 standard deviation window. Here, the boundary of the pupil is discernable visually, but the outer boundary of the iris is noticeably weaker than that of the pupil. In addition, neighborhoods about the eyelashes show particularly strong values. Using empirical techniques, a global threshold is applied to this image to create a binary image, and in the binary image the pupil stands out well as a circle of unknown radius. We assume a circular pupil so that a circular Hough transform readily provides the location of the center of the pupil and its radius.

The harder problem is determining the outer boundary of the iris, known as the limbic boundary. The transition from the iris to the sclera is more gradual than the transition from the pupil to the iris, and the iris itself may already be very light. Moreover, there can be eyelashes and/or eyelids obscuring portions of the boundary. Therefore, to find the limbic boundary, the pupil center is often used as a starting point to find the limbic circle center. But since the pupil center is sometimes closer to the nose than the limbic center, an adjustment must enter into the method as well. As it turned out, using the standard deviation window with a global threshold is not a robust means for detecting the limbic boundary. In addition, the local skewness did not empirically reveal a distinct limbic boundary. However, we have found that the local kurtosis works well.

Figure 2: Local Standard Deviation (3x3) of the Iris in Fig. 1.

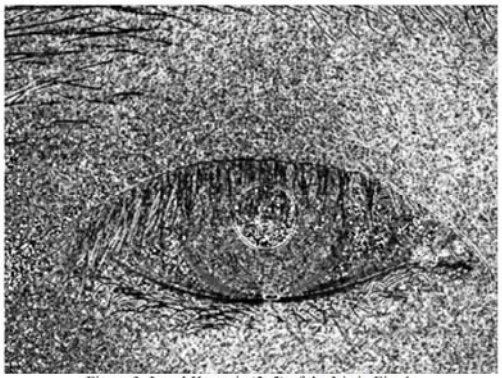

Figure 3: Local Kurtosis (5x5) of the Iris in Fig. 1.

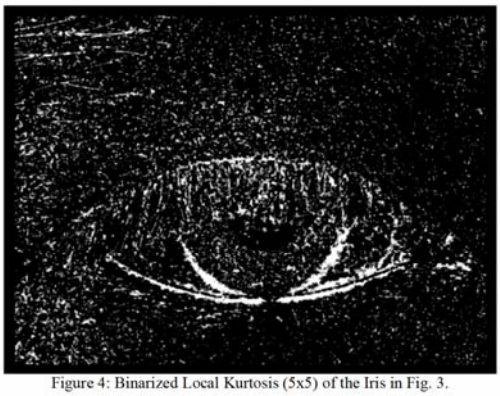

Figure 4: Binarized Local Kurtosis (5x5) of the Iris in Fig. 3.

To determine the limbic boundary on the iris of Fig. 1, we first compute the local kurtosis in 5x5 neighborhoods of each pixel. In the resulting image (see Fig. 3), we observe that the values near the limbic boundary are relatively stable in small neighborhoods, compared with the "salt and pepper" appearance in the sclera and in the interior of the iris. Therefore to create the binary image (see Fig. 4), we assign a value of 1 (white) to locations for which kurtosis values within a 5x5 neighborhood are nearly constant. Next, we consider a few possible centers for the limbic boundary: the pupil center and the pupil center shifted by small increments slightly to the left or right, for right vs. left eyes respectively. For each center, we generate a family of overlapping concentric annuli, all with the same "thickness" (meaning the outer radius minus the inner radius), and with the radii increasing in small increments. The limbic boundary shown in Fig. 5 is simply the annulus containing the highest percentage of 1s when "overlaid" with the image in Fig. 4, maximized over all the annuli corresponding to the various centers and radii.

A few remarks are in order. First, the neighborhood sizes, the definition of "nearly constant", the annuli thickness, the choice of increments of the radii and the increments for the adjustment of the circle center are all of course somewhat arbitrary. Moreover, the exact location of the limbic boundary is subjective. For these reasons we make no claim that we have determined the optimal parameters for every user or database. This is actually an advantage, since the parameters can be easily adjusted to prioritize speed or precision, and adjusted according to the properties of a particular camera. Also, the algorithm appears quite robust: even when eyelashes and eyelids obscure the top and bottom of the iris, the limbic boundary arcs in Fig. 3 are strong enough to ensure that a good annulus is easy to identify.

VI. RESULTS

The segmentation algorithm was run on a number of images in the University of Bath iris database, as well as the Naval Academy iris database. In almost all cases, as long as the pupil could be located, the limbic boundary could be found. In a small number of images from the Naval Academy database, an unusually light iris color and very gradual iris boundary made it difficult to find the limbic boundary. In these cases, it may be possible that a local threshold rather than a global threshold could detect a weak limbic boundary.

The effect of a pupil and iris that are not concentric is illustrated in Figs. 7-8. In Fig. 7, we assumed the center of the iris was collocated with the center of the pupil, and searched for the best-fit annulus to detect the limbic boundary. Here, the best-fit annulus included the entire left side of the iris, while the right side is overestimated such that a portion of the sclera is segmented. To account for the pupil and iris centers not being collocated, the search for the best-fit annulus includes steps to offset the center of the annulus to the left and right to see if a better match can be found (this was mentioned in Section III). For the image in Fig. 7, an offset of the center of the annuli to the left resulted in a better fit for the limbic boundary, which is displayed in Fig. 8.

As another example, Fig. 9 is an iris image from the U.S.

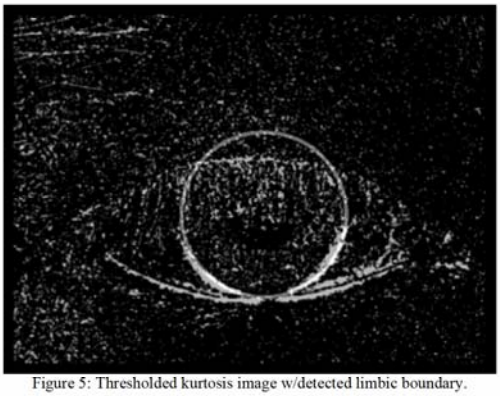

Figure 5: Thresholded kurtosis image w/detected limbic boundary.

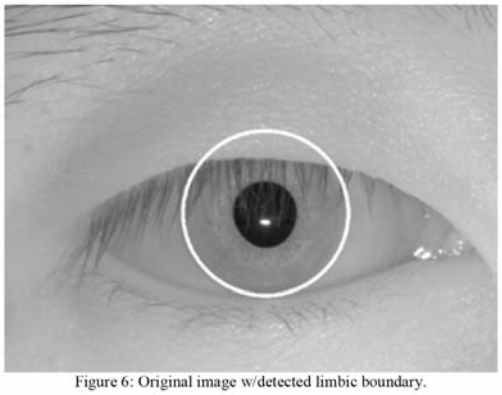

Figure 6: Original image w/detected limbic boundary.

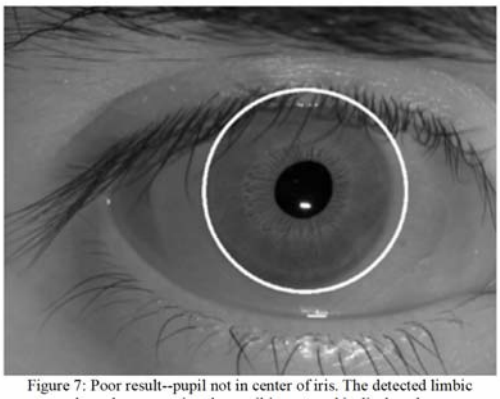

Figure 7: Poor result--pupil not in center of iris. The detected limbic boundary assuming the pupil is centered is displayed.

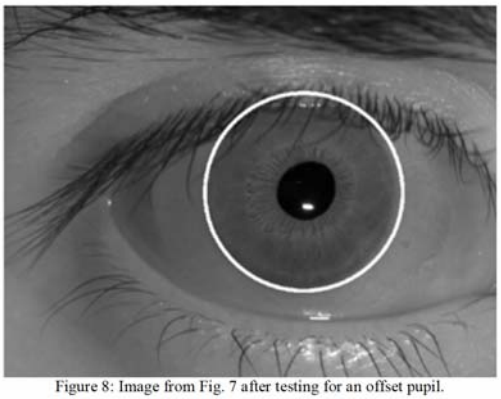

Figure 8: Image from Fig. 7 after testing for an offset pupil.

Naval Academy database. Figure 10 is the local kurtosis image for this iris, and Fig. 11 shows the detected limbic boundary. This is an image of a left eye, and the pupil is offset within the iris towards the nose (to the left in this image). The best-fit limbic boundary again resulted after a shift from the center of the pupil.

VII. CONCLUSIONS

A new algorithm for iris segmentation was presented that incorporates local statistical analysis. The pupil is easily found using a small local standard deviation window, thresholding and then using the Hough transform to find the circular pupil. The center of the pupil and its radius are then used to find the outer boundary of the iris. The outer boundary is typically more challenging because there is a gradual transition in gray values in the NIR iris image from the outer edge of the iris to the sclera, such that a sharp edge is not present. Here, the local kurtosis has proven its ability to distinguish the limbic boundary, even when the iris is occluded. Kurtosis is a measure of how the peaked a distribution of values is. In general, around the limbic boundary, the variation in the kurtosis is smaller than inside the iris itself or inside the sclera, and this gives a visual indication of the location of the boundary.

This method carries with it several assumptions. First, it is assumed that the iris images are orthogonal, such that the eye is looking directly at the camera. In conjunction with this, it is assumed that the pupil and the limbic boundary of the iris is circular, which is not always accurate. Finally, the method assumes that the boundary between the iris and the sclera (the limbic boundary) is not so gradual that the local statistics (here, the kurtosis) across the boundary cannot be differentiated.

Further testing on additional imagery is in progress. In addition, further research in the use of this method on non-orthogonal (off-axis) images is in progress. In this case, a search for elliptical-shaped boundaries is more appropriate.

REFERENCES

[1] J. Huang, Y. Wang, T. Tan, and J. Cui, "A new iris segmentation method for recognition," *Proceedings of the 17th International Conference on Pattern Recognition, 2004 (ICPR 2004)*, Vol. 3, pp. 554-557, Aug. 2004.

[2] W. Kong and D. Zhang, "Accurate iris segmentation based on novel reflection and eyelash detection model," *Proceedings of the 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing*, pp. 263-266, May 2001.

[3] Q. Tian, Q. Pan, Y. Cheng and Q. Gao, "Fast algorithm and application of Hough transform in iris segmentation," *Proceedings of the 2004 International Conference on Machine Learning and Cybernetics*, Vol. 7, pp. 3977-3980, Aug. 2004.

[4] A. Bachoo and J. Tapamo, "A segmentation method to improve iris-based person identification," Proceedings of the 7th AFRICON Conference, Vol. 1, pp. 403-408, Sept.

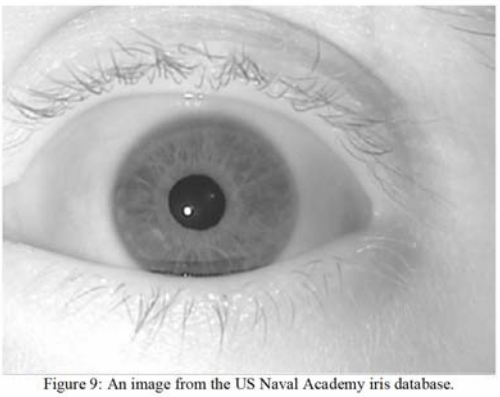

Figure 9: An image from the US Naval Academy iris database.

2004.

[5] J. Daugman, "High Confidence Visual Recognition of Persons by a Test of Statistical Independence", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 11, pp. 1148-1161, 1993.

[6] University of Bath, U.K. Iris Image Database, http://www.bath.ac.uk/elec-eng/pages/sipg/irisweb.

[7] R.P. Wildes, J.C. Asmuth, G.L. Green, S.C. Hsu, R.J. Kolczynski, J.R. Matey, and S.E. McBride, "A Machine Vision System for Iris Recognition", Mach. Vision Application, Vol. 9, pp.1-8, 1996.

[8] W.W. Boles and B. Boashash, "A Human Identification Technique Using Images of the Iris and Wavelet Transform", *IEEE Transactions on Signal Processing*, Vol. 46, No. 4, pp. 1185-1188, 1998.

[9] J.E. Siedlarz, "Iris: More detailed than a fingerprint", *IEEE Spectrum*, vol. 31, pp. 27, 1994.

[10] Gonzalez, R.C. and Woods, R.E. *Digital Image Processing*. Reading, MA: Addison-Wesley Publishing Co., 1992.

[11] Y. Du, C.-I. Chang, H. Ren, F.M. D'Amico, J. Jensen, J., "A New Hyperspectral Discrimination Measure for Spectral Similarity", *Optical Engineering*, Vol. 43, No. 8, 2004.

[12] Y. Du, R. Ives, D. Etter, T. Welch, and C.-I Chang, "A One-Dimensional Approach for Iris Identification", presented at SPIE Aerospace/Defense Sensing, Simulation, and Controls, Orlando, FL, April, 2004.

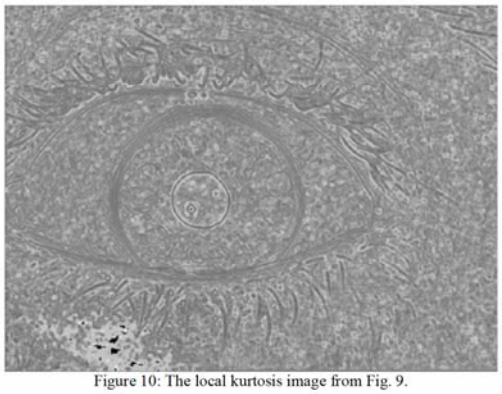

Figure 10: The local kurtosis image from Fig. 9.

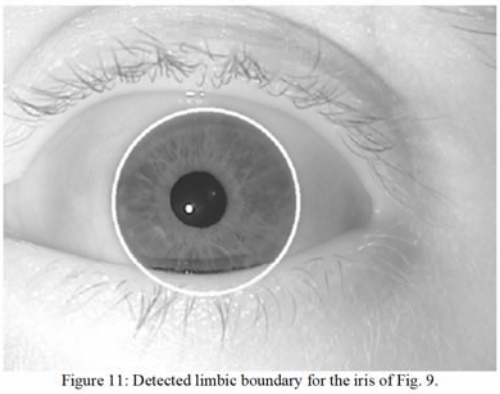

Figure 11: Detected limbic boundary for the iris of Fig. 9.